# Sistema de recomanació de problemes per Jutge.org

Treball de final de grau
Octubre 2016

**Autor** Miquel Rius
**Director** Jordi Petit
**Especialitat** Computació
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC) – BarcelonaTech

**Resum**

El jutge de programació en línia de la UPC, Jutge.org, ha estat acumulant dades des de la seva creació al 2006. El tractament d'aquestes dades pot millorar l'efectivitat docent i millorar les característiques que ofereix Jutge.org als seus usuaris.

En aquest projecte proposem tres mètodes de calcul de la dificultat dels problemes i l'experiència dels usuaris. Un mètode senzill que utilitzem com a base per als altres dos. Un de basat en les relacions entre els usuaris i els problemes. I un basat en SVD.

També entrenem un SVM amb els logs de Jutge.org per predir l'abandonament dels cursos i la nota final dels estudiants. Finalment, construïm un recomanador de problemes utilitzant les mesures de la dificultat.

**Abstract**

UPCs online programming judge, Jutge.org, has been gathering user data since its creation in 2006. Processing that data could improve teaching effectiveness and the tools Jutge.org offers its users.

In this project we propose three methods for calculating problem difficulty and user experience. One simple which we use as baseline for comparisons. One which is based on the relations between users and problems. And one that uses SVD.

We also train a SVM with Jutge.org logs for predicting student dropout and their final notes. Lastly, we present a problem recommender that uses the problem difficulty.

# Índex


**1 Introducció** **4**
- 1.1 Objectius . . . 4
- 1.2 Actors implicats . . . 4
- 1.3 Estat de l'art . . . 5
- 1.4 Abast . . . 8

**2 Planificació** **9**
- 2.1 Tasques . . . 9
- 2.2 Taula de temps . . . 11
- 2.3 Diagrama de Gantt . . . 12
- 2.4 Desviació de la planificació original . . . 12

**3 Metodologia** **14**

**4 Pressupost** **16**
- 4.1 Recursos humans . . . 16
- 4.2 Recursos hardware . . . 17
- 4.3 Costos indirectes . . . 17
- 4.4 Recursos software . . . 17
- 4.5 Desviació del pla original . . . 18
- 4.6 Resum . . . 19

**5 Informe de sostenibilitat** **19**
- 5.1 Viabilitat econòmica . . . 19
- 5.2 Viabilitat social . . . 19
- 5.3 Viabilitat ambiental . . . 19
- 5.4 Taula de sostenibilitat . . . 20

**6 Descripció de les dades** **21**
- 6.1 Descripció de la base dades . . . 21
- 6.2 Descripció dels cursos . . . 26
- 6.3 Descripció dels logs d'accés . . . 28
- 6.4 Detalls sobre la confidencialitat de les dades . . . 29

**7 Anàlisi de la dificultat dels problemes i l'experiència dels usuaris** **30**
- 7.1 Descripció del problema . . . 30
- 7.2 Algoritmes . . . 30
- 7.3 Validació . . . 36

**8 Predicció d'abandonament i nota final dels estudiants** **43**
- 8.1 Descripció del problema . . . 43
- 8.2 Descripció de la informació extreta . . . 43
- 8.3 Predicció d'abandonament . . . 46
- 8.4 Predicció de la nota final . . . 50

8.5 Valoració dels resultats obtinguts . . . . . 54

**9 Creació d'un recomanador** **55**
9.1 Descripció del problema . . . . . 55
9.2 Descripció del recomanador . . . . . 55
9.3 Validació . . . . . 57

**10 Conclusions** **59**

# 1 Introducció

Gràcies a Internet molts cursos ofereixen recursos en línia per als estudiants, des de material complementari al curs fins a qüestionaris per auto-avaluar-se i practicar els coneixements assolits. En l'àmbit de la programació existeixen uns sistemes anomenats jutges en línia.

Aquests jutges són capaços de compilar, executar i avaluar, mitjançant uns jocs de proves, els programes enviats pels seus usuaris. Normalment es disposa d'un petit joc de proves públic que serveix per mostrar el funcionament esperat del programa. Per altra banda existeix un joc de proves privat molt més gran que serveix per decidir si la solució proposada és correcte. L'avaluació es realitza en un entorn segur per prevenir danys contra el servei ja siguin accidentals o intencionats.

La Universitat Politècnica de Catalunya (UPC) i, més particularment, la Facultat d'Informàtica de Barcelona (FIB) i la Facultat de Matemàtiques i Estadística (FME) utilitzen un jutge en línia en alguns dels seus cursos: Jutge.org [25].

Jutge.org és un jutge en línia creat i mantingut per la UPC. Entra en funcionament en 2006 i des de aleshores continua rebent estudiants de la UPC en el seu camí cap la titulació.

Durant els 10 anys des de la seva creació Jutge.org ha estat acumulant dades. L'estudi i tractament d'aquestes dades pot aportar nova informació als seus usuaris: estudiants i professors. Aquest projecte intenta extreure aquesta informació.

## 1.1 Objectius

L'objectiu principal d'aquest projecte és dissenyar, implementar i analitzar mètodes automàtics per extreure informació de les dades de Jutge.org. En particular, es vol:

1. Mesurar la dificultat dels problemes i l'experiència dels usuaris.
2. Predir els alumnes que poden abandonar un curs donat.
3. Crear un sistema intel·ligent de recomanacions de problemes.

Això s'aconseguirà aplicant tècniques d'aprenentatge automàtic i mineria de dades. Aquest objectius es tradueixen en les tasques que es detallen a la secció 2.1.

## 1.2 Actors implicats

Aquest treball involucra diverses persones i entitats:

**Desenvolupador:** Aquest treball té un únic desenvolupador, jo, Miquel Rius.

**Director:** El director d'aquest treball és Jordi Petit, professor de la FIB i la FME. També és un dels creadors i l'administrador actual de Jutge.org.

**Jutge.org:** Jutge.org aporta totes les dades que s'analitzaran en aquest projecte i, finalment, aprofitarà el software creat.

**Estudiants de la UPC:** Els estudiants, que són els principals usuaris de Jutge.org, són actors en tant que s'analitzen les seves dades passades com que rebran els resultats de l'anàlisi de les dades d'aquest projecte.

**Professors de la UPC:** El professorat de la UPC podrà rebre/utilitzar el software per tal de millorar la seva activitat docent.

## 1.3 Estat de l'art

En aquesta secció es descriu l'estat de l'art dels jutges de programació, l'aprenentatge automàtic, les tècniques de detecció d'abandonament, els sistemes de recomanació i les tècniques de clústering. Finalment es parla del llenguatge de programació Julia.

### 1.3.1 Jutges de programació

Els Jutges de programació són serveis en línia creats per a la correcció automàtica de problemes de programació. S'utilitzen principalment en concursos de programació, p.e ACM-ICPC, IOI, TopCoder.

Aquests jutges disposen de problemes per als seus usuaris i de petits jocs de proves o exemples per demostrar el correcte funcionament de la solució al problema proposat. Els usuaris envien el codi de la seva solució i el jutge el compila i l'executa en un entorn segur. Els Jutges disposen de jocs de prova privats per avaluar les solucions proposades: si una solució passa tots els jocs de prova s'avalua correctament, si no, incorrectament.

Hi ha molts jutges en línia de programació; a continuació se'n referencien alguns i s'esmenten algunes de les característiques i mancances relaciones amb el treball:

- timus.ru [32]: Un jutge de programació que calcula la dificultat dels problemes com el nombre de nous usuaris que el resolen per unitat de temps [10]. No implementa recomanador.

- codeabbey [4]: Es defineix com la abadia dels programador i per això el sistema té temàtica eclesiàstica. La dificultat dels problemes, "blessing", es calcula com el nombre d'usuaris que l'han resolt [33]. No implementa recomanador.

- ProjectEuler [27]: Un jutge de problemes matemàtics per a tothom que vulgui mantenir la ment ocupada. No té recomanador ni dificultat, però mostra el número de persones que l'han resolt.

Les tècniques que apliquen els altres jutges de programació per calcular la dificultat, principalment comptar el nombre d'usuaris que els resolen, no tenen

en compte que la dificultat del problema depèn de l'usuari que el resol, és a dir, usuaris més experimentats consideren els problemes com a més senzills que usuaris amb menys experiència.

El nostre sistema, Jutge.org, no implementa càlcul de dificultat ni recomanador. Considerem incomplets els sistemes de càlcul de dificultat dels anteriors jutges i no trobem sistemes de recomanació. Per això en aquest treball s'implementarà una solució pròpia.

### 1.3.2 Aprenentatge automàtic

L'aprenentatge automàtic és el camp d'estudi que dóna als ordinadors l'habilitat d'aprendre sense ser explícitament programats [19]. Consisteix en la creació d'un model a partir d'un conjunt d'exemples anomenat conjunt d'entrenament (training). Aquest model serà utilitzat per donar respostes a les entrades en lloc de seguir una programació estricte i estàtica. Els algoritmes d'aprenentatge automàtic *s'entrenen* amb aquest conjunt d'entrenament per crear el model.

Com que els models depenen de les dades amb les que s'ha entrenat poden donar resultats diferents. Per avaluar la precisió d'aquests models s'utilitza un segon conjunt anomenat de test o de validació. *Durant la creació del model, l'algoritme no ha vist les dades d'aquest conjunt de test* i per tant es poden considerar dades reals.

Normalment les dades de prova són limitades, i per fer bons models es necessita utilitzar la major quantitat possible de dades per a entrenar. Però si el conjunt de test és massa petit no s'aconsegueix mesurar bé la precisió del model. Una solució és utilitzar una tècnica anomenada $k$-fold cross-validation, mostrat a la figura 1. Això permet utilitzar una proporció $(k-1)/k$ de les dades per entrenar i utilitzar totes les dades per avaluar la precisió [2].

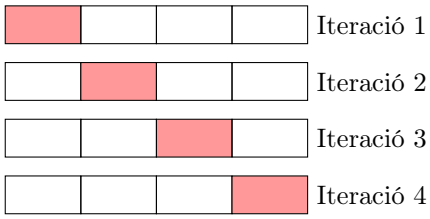


Figura 1: $k$-fold cross-validation per a $k = 4$. El conjunt de dades inicial es divideix en $k$ grups d'igual mida. Després $k-1$ grups són utilitzats com a conjunt d'entrenament per entrenar uns models que són avaluats contra el grup que falta (conjunt de test), indicat pel requadre vermell. Això es repeteix fins que tots els grups han estat conjunt de test. Cada una de les $k$ iteracions dona una precisió i finalment es calcula la seva mitjana.

### 1.3.3 Detecció d'abandonament

En ocasions els estudiants abandonen un curs. A les universitats i altres institucions acadèmiques els interessa detectar amb antelació aquest casos d'abandonament. Conèixer que un estudiant abandonarà un curs abans que això passi permetria poder donar un suport addicional a aquest estudiant i evitar l'abandó.

Alguns sistemes de detecció utilitzen el currículum dels estudiants per fer aquesta predicció [6].

Altres propostes, utilitzades als cursos en línia massius (MOOC), no disposen del currículum de l'estudiant però sí que disposen de la interacció d'aquest amb el curs. Examinen la tendència dels estudiants utilitzant el curs fins identificar els patrons que comparteixen aquells que l'abandonen [16].

### 1.3.4 Sistemes de recomanació

No hi ha jutges de programació que implementin sistemes de recomanació però sí que hi ha altres serveis que els implementen.

Amb la expansió d'Internet cada cop hi ha més informació i cada cop costa més trobar una informació concreta. Els sistemes de recomanació s'utilitzen com a filtres d'informació que donen bons resultats. Amazon s'aprofita dels filtres col·laboratius per recomanar els seus productes als clients [18].

Hi ha dos tipus de sistemes de recomanació: els basats en el contingut i els filtres col·laboratius (CF) sent aquest últim el més utilitzat en la actualitat [9].

La assumpció darrere dels CF és que les valoracions dels usuaris poden ser seleccionades i agregades de tal forma que proporcionin una raonable predicció de les preferències de l'usuari actiu, és a dir, s'assumeix que si un grup d'usuaris coincideixen en la valoració d'un conjunt d'objectes llavors també coincidiran en la valoració d'altres objectes [22].

Un cop tinguem la dificultat dels problemes estimada, considerem que es podrà utilitzar com a valor de similitud entre els problemes i construir una versió modificada d'un CF, com el descrit a [29].

### 1.3.5 Clústering

Clústering representa la tasca d'agrupar una sèrie d'observacions en grups o clústers de manera que els elements d'un mateix clúster són més semblants entre ells que entre elements d'altres clústers.

Depenent de la definició de "semblants" apareixen diferents nocions de clústers i amb elles diferents algoritmes. En els models de distribució els clústers són modelats a partir de distribucions estadístiques, com a, per exemple, la suma de Gausianas [2].

En els models de densitat són clúster les regions més denses de l'espai.

Un dels algoritmes de clústering més utilitzats, considerat com el segon algoritme més important a data mining [34], és l'algoritme $k$-means[17]. Aquest algoritme basat en centres pren les observacions com a punts en un espai euclidià multidimensional i forma clústers minimitzant les distancies dels elements d'un mateix clúster al "centre del clúster" [2].

### 1.3.6 Julia

Julia [31] és un llenguatge de programació jove, la primera versió va aparèixer al 2012, que pretén unir la simplicitat dels llenguatges dinàmics amb la velocitat dels llenguatges d'alt rendiment.

L'objectiu de Julia és aconseguir un rendiment similar a C o FORTRAN proporcionant la llibertat d'expressió en operacions numèriques que tenen llenguatge dinàmics com R, MATLAB o Octave.

Julia és un llenguatge pensat per a les àrees tècniques i per això serà utilitzat en aquest projecte.

## 1.4 Abast

Aquest projecte es centrarà *exclusivament* en la base de dades de Jutge.org, definida al apartat 6.1, i en els seus logs d'accés, descrits al apartat 6.3. S'aplicaran tècniques d'aprenentatge automàtic per extreure la informació desitjada.

Diverses tècniques utilitzen preguntes als usuaris per tal de extreure aquesta informació, la més important la dificultat [28, 15]. Un sistema així hauria de ser implementat, a més trigaria una considerable quantitat de temps en començar a donar dades útils. No disposem d'aquest temps i per això hem decidit deixar-lo fora de l'abast del treball.

El flux del programa i el nombre de línies de codi de les solucions sembla tenir relació amb la dificultat del problema [1]. Els experiments aportats per [28] corroboren aquest fet. Nosaltres disposem de la solució dels problemes i podríem utilitzar-los però hem decidit excloure-les d'aquest estudi.

Finalment els algoritmes desenvolupats no s'integraran a Jutge.org. El treball d'implementació del codi a Jutge.org i interacció amb els usuaris (interfície d'usuari) no forma part d'aquest treball i queda pendent com a treball futur.

# 2 Planificació

Originalment, es va planificar que el projecte durés quatre mesos, des de febrer fins a maig. Pels motius descrits a l'apartat 2.4 la planificació original no va poder ser obeïda. Els canvis queden reflectits en l'apartat 2.4.

En aquesta secció es detallen les tasques del projecte, les desviacions i els diagrames de Gantt.

## 2.1 Tasques

Aquest apartat descriu les tasques dels projecte, amb els seus temps previstos, requeriments i possibles complicacions. La taula 1 resumeix aquesta secció.

Com que el projecte és farà en Julia i l'autor n'ha d'aprendre totes les tasques tenen el risc de patir petits contratemps.

### 2.1.1 Fita inicial

Aquesta és la etapa inicial del projecte, GEP. On es planifica el projecte i s'encamina cap a la seva resolució.

| | |
|---|---|
| Temps previst | 75h |
| Requeriments | Cap |
| Possibles complicacions | Cap |

### 2.1.2 Preparació del entorn de treball

Instal·lar totes les eines necessàries i preparar l'ordinador.

| | |
|---|---|
| Temps previst | 5h |
| Requeriments | Cap |
| Possibles complicacions | Cap |

### 2.1.3 Familiaritzar-se amb les dades

Les dades es troben en una base de dades estructurada d'una manera concreta, per tal de poder extreure informació d'ella és important conèixer quins camps disposa aquesta base de dades.

| | |
|---|---|
| Temps previst | 5h |
| Requeriments | Cap |
| Possibles complicacions | Cap |

### 2.1.4 Estimació de la dificultat dels problemes i l'experiència dels usuaris

Aquesta tasca es centra en inferir la dificultat dels problemes a partir de les dades. És la tasca en la que més temps es dedicarà perquè creiem que és la

tasca més difícil del projecte.

| | |
|---|---|
| Temps previst | 120h |
| Requeriments | Tenir el sistema preparat i conèixer les dades |
| Possibles complicacions | Mesurar la dificultat pot necessitar criteris més complexos que els que estem mesurant i pot donar resultats de inferior qualitat a la esperada. |

#### 2.1.5 Agrupació dels problemes en clústers

Crear un sistema d'agrupació de problemes. Estimem que és bona idea crear un sistema automàtic que agrupi els problemes. Aquestes agrupacions poden simplificar la tasca de crear un recomanador.

| | |
|---|---|
| Temps previst | 50h |
| Requeriments | Tenir el sistema preparat i conèixer les dades |
| Possibles complicacions | Cap |

#### 2.1.6 Creació del sistema de recomanacions

Aquesta tasca es dedicarà a la creació del sistema de recomanacions.

| | |
|---|---|
| Temps previst | 75h |
| Requeriments | Estimació de la dificultat i clústers de problemes |
| Possibles complicacions | Si la dificultat estimada o els clústers no són prou bons el recomanador no podrà ser-ho. |

#### 2.1.7 Predicció d'abandonament

Aquesta tasca *no formava part* de la planificació original (veure l'apartat 2.4).

Aquesta tasca consisteix en la creació d'un predictor dels estudiants que abandonen el curs. I d'un predictor per a les notes dels estudiants

| | |
|---|---|
| Temps previst | 90h |
| Requeriments | Cap |
| Possibles complicacions | Cap |

#### 2.1.8 Redacció de la memòria i documentació

Aquesta etapa es dedicarà a la redacció d'un document que expliqui tot el treball que s'ha fet.

| | |
|---|---|
| Temps previst | 40h |
| Requeriments | El document s'anirà redactant a mesura que es completin les tasques. |
| Possibles complicacions | El document es redactarà en LaTeX i l'autor n'ha d'aprendre. |

## 2.2 Taula de temps

S'espera un total de 370 hores; la duració del projecte són 16 setmanes així que es correspon a unes 23 hores setmanals. Si no hi ha complicacions s'hauria de poder complir el termini.

| Tasca | Temps (h) |
|---|---|
| Fita Inicial | 75 |
| Preparació del entorn de treball | 5 |
| Familiaritzar-se amb les dades | 5 |
| Estimació dificultat i experiència | |
| Disseny del algoritme | 48 |
| Implementació | 24 |
| Valoració | 48 |
| Agrupació dels problemes en clústers | |
| Disseny del algoritme | 15 |
| Implementació | 10 |
| Valoració | 25 |
| Creació del recomanador | |
| Disseny del algoritme | 30 |
| Implementació | 15 |
| Valoració | 30 |
| Documentació | 40 |
| Total | 370 |

Taula 1: Resum del temps de cada tasca.

## 2.3 Diagrama de Gantt

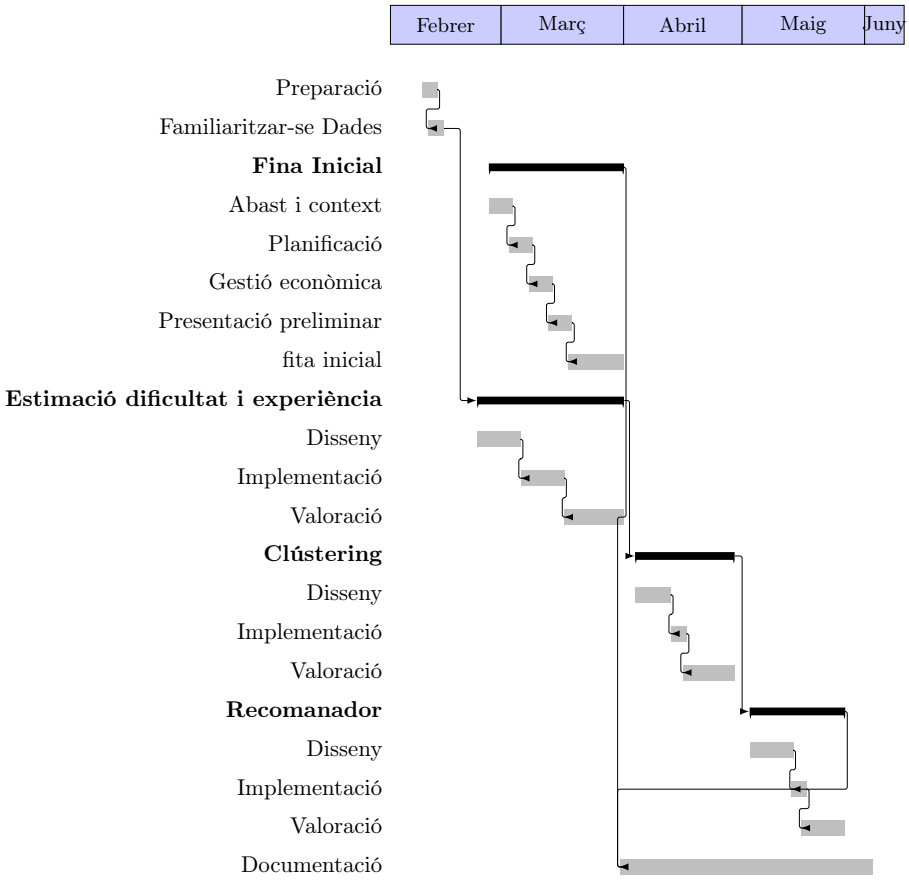


Figura 2: Diagrama de Gantt (Planificació).

## 2.4 Desviació de la planificació original

Per un excés de feina per part d'altres assignatures que afectava l'autor d'aquest projecte, una mala gestió per part d'aquest i diverses complicacions en el transcurs del projecte, la planificació original del projecte no ha pogut ser obeïda.

Les expectatives originals del autor han resultat ser massa optimistes i en un intent de millorar la qualitat de l'estimador de dificultat s'ha invertit massa temps.

Finalment la duració del projecte ha estat de set mesos, des de febrer fins a

setembre.

Les principals diferències amb la planificació original són:

- La versió de Julia dels repositoris és una versió molt antiga i això estava provocant problemes de compatibilitat amb algunes llibreries. Finalment l'autor del projecte va decidir instal·lar Julia compilant-lo des de codi, va ser lent però es van resoldre els problemes.

- El clústering no es va fer en favor d'utilitzar les agrupacions pròpies de Jutge.org: els cursos.

  Considerem que les característiques natives d'aquests grups els fan perfectes. Per una banda es tracta d'agrupacions molt reduïdes, això soluciona el problema de la mida de les dades i el de la dispersió d'aquestes.

  Per altra banda han estat creats per professors com a material docent i per tant els problemes que contenen tenen un nivell similar o comparteixen alguna característica que els fa rellevants per a les classes impartides. A més aquest cursos acostumen a ser privats i el professor ha d'invitar als seus alumnes, això fa que tots els alumnes d'un curs tinguin un nivell similar.

- És va afegir una nova tasca: Predicció d'abandonament del estudiants d'un curs.

La taula següent conté l'actualització de la taula 1 i la figura 3 conté el diagrama de Gantt actualitzat.

| Tasca | Temps previst (h) | Temps real (h) |
|---|---|---|
| Fita Inicial | 75 | 75 |
| Preparació del entorn de treball | 5 | 8 |
| Familiaritzar-se amb les dades | 5 | 5 |
| Estimació dificultat i experiència | | |
| Disseny del algoritme | 48 | 90 |
| Implementació | 24 | 40 |
| Valoració | 48 | 50 |
| Agrupació dels problemes | | 5 |
| Disseny del algoritme | 15 | - |
| Implementació | 10 | - |
| Valoració | 25 | - |
| Creació del recomanador | | |
| Disseny del algoritme | 30 | 20 |
| Implementació | 15 | 8 |
| Valoració | 30 | 15 |
| Predicció d'abandonament | | |
| Disseny del algoritme | - | 45 |
| Implementació | - | 15 |
| Valoració | - | 30 |
| Documentació | 40 | 65 |
| Total | 370 | 471 |

Taula 2: Resum dels temps reals de cada tasca en comparació amb els previstos a la taula 1.

# 3 Metodologia

Aquest projecte comença per extreure informació de les dades i continua per analitzar aquesta informació. Però es desconeix com és aquesta informació a priori. Per això es va optar per una metodologia que fos capaç d'adaptar-se i canviar ràpidament: Scrum [30].

Scrum és una metodologia àgil, iterativa i interactiva per al desenvolupament de software inspirada en el procés empíric. Accepta que els requeriments d'un projecte poden canviar durant el desenvolupament d'aquest i intenta ser ràpid en la adaptació als canvis.

En la metodologia Scrum es selecciona un subconjunt dels requeriments del projecte i es fa un "sprint" d'una durada determinada. Al final d'aquest sprint es determina en quina mesura s'han completat els objectius del sprint i que es pot millorar. D'aquesta manera la completesa del producte va augmentant gradualment.

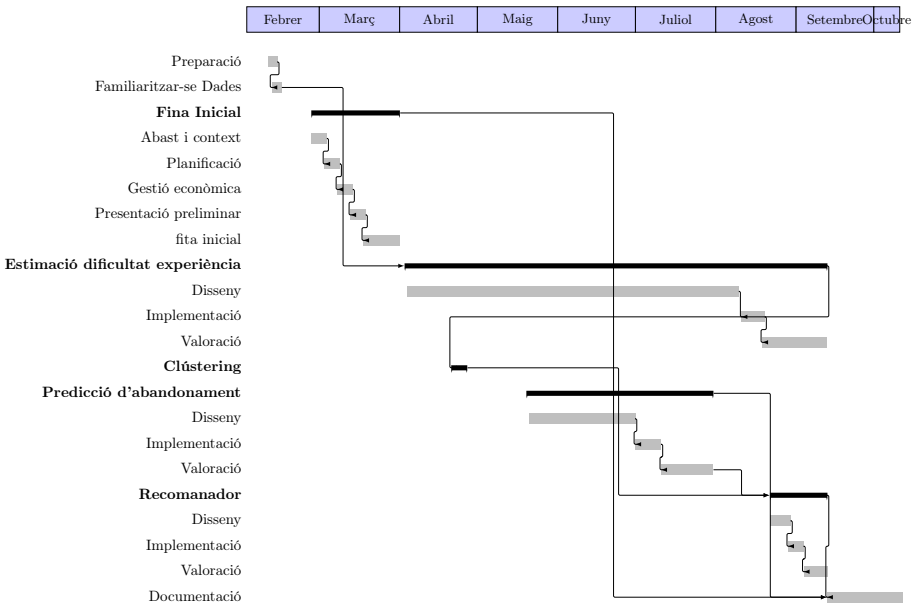


Figura 3: Diagrama de Gantt revisat.

Cada tasca del projecte comença amb una reunió amb el director on es defineix un sprint d'una setmana. Passada aquesta setmana hi ha una nova reunió per avaluar els resultats.

A continuació farem un esbós de com s'ha fet el treball.

- Primer es comença per un anàlisis de la base de dades per tal de familiaritzar-se amb aquestes i veure com són. Durant aquesta fase també s'indentificaran o descartaran dades que no siguin desitjables, com per exemple les d'un usuari que no ha resolt cap problema.
- Amb les dades ja analitzades i preparades es seleccionen uns subconjunts per als que disposem d'informació. Aquestes dades són les de training i validació per entrenar i verificar els algoritmes.
- S'han dissenyat i implementat diferents algoritmes que calculin la dificultat dels problemes i la experiència dels usuaris i es comparen els resultats amb els conjunts de validació. Diferents indicadors avaluen els algoritmes: temps d'execució, qualitat de la solució, cost d'implementació ... Es selecciona el més convenient.
- Es dissenya, implementa i avalua amb el conjunt de validació, un mètode capaç de predir l'abandonament d'un curs.

- Amb la dificultat i l'experiència dels usuaris s'implementa un sistema de recomanacions.

# 4 Pressupost

En aquesta secció es descriu el pressupost del treball i el seu cost real. La taula 9 a l'apartat 4.6 conté un resum dels costos del projecte.

## 4.1 Recursos humans

La taula 3 descriu el salaris del personal i la taula 4 conté el cost personal per a cada tasca.

| Personal | | Salari per hora |
|---|---|---|
| Cap de projecte | (C) | 75 € |
| Dissenyador | (D) | 60 € |
| Desenvolupador | (P) | 27 € |
| Tester | (T) | 30 € |

Taula 3: Pressupost de recursos humans.

| Tasca | Temps (h) | Responsable | Cost |
|---|---|---|---|
| Fita Inicial | 75 | C | 5625 € |
| Preparació del entorn de treball | 5 | P | 135 € |
| Familiaritzar-se amb les dades | 5 | P | 135 € |
| Estimació dificultat i experiència | | | |
| Disseny del algoritme | 48 | D | 2880 € |
| Implementació | 24 | P | 648 € |
| Valoració | 48 | T | 1440 € |
| Agrupació dels problemes | | | |
| Disseny del algoritme | 15 | D | 900 € |
| Implementació | 10 | P | 270 € |
| Valoració | 25 | T | 750 € |
| Creació del recomanador | | | |
| Disseny del algoritme | 30 | D | 1800 € |
| Implementació | 15 | P | 405 € |
| Valoració | 30 | T | 900 € |
| Documentació | 40 | C | 3000 € |
| Total | | | 18888 € |

Taula 4: Pressupost de personal en funció de cada tasca.

## 4.2 Recursos hardware

Aquest treball es farà en un ordinador portàtil i utilitzant un Raspberry pi com a servidor del repositori del treball.

| Producte | Preu | Vida Útil (anys) | Amortització |
|---|---|---|---|
| Portàtil Toshiba | 400 € | 5 | 38,67 € |
| Raspberry Pi B | 25 € | 5 | 3,25 € |
| Total | 425 € | | 41,92 € |

Taula 5: Pressupost en Hardware.

## 4.3 Costos indirectes

Aquest projecte també té una despesa elèctrica per fer funcionar el hardware. La taula 6 mostra el cost elèctric estimat. S'assumeix un cost de 0.18 €/kWh[26].

| Hardware | Consum (W/h) | Temps estimat (h) | Cost |
|---|---|---|---|
| Portàtil Toshiba | 300 | 370 | 19,98 € |
| Raspberry Pi B | 3,5 | 24h × 30d × 4m | 1,81 € |
| Total | | | 21,79 € |

Taula 6: Pressupost per al cost elèctric.

## 4.4 Recursos software

S'ha optat per realitzar aquest treball amb eines de software lliure que poden ser utilitzades gratuïtament:

| | |
|---|---|
| Ubuntu | Sistema Operatiu |
| Raspbian | Sistema Operatiu |
| PostgreSQL | Base de dades |
| Julia | Llenguatge de programació |
| LaTeX | Sistema de composició de documents |
| git | Sistema de control de versió distribuït |
| atom | Editor de text |
| vim | Editor de text |
| gawk | Implementació de GNU del llenguatge per processat de text, AWK |
| Zotero | Gestor de referències |
| gnuplot | Generador de gràfiques |

Per tant aquest projecte no té cost en software.

## 4.5 Desviació del pla original

Per les raons establertes a la secció 2.4 el cost real d'aquest projecte ha resultat superior al previst.

S'han treball més hores i això ha incrementat el cost en recursos humans en 6189 € i els costos indirectes en 6,81 €. Els detalls és troben descrits a les taules 7 i 8.

### 4.5.1 Recursos humans

| Tasca | Temps previst (h) | Temps real (h) | Responsable | Cost previst | Cost real |
|---|---|---|---|---|---|
| Fita Inicial | 75 | 75 | C | 5625 € | 5625 € |
| Preparació del entorn | 5 | 8 | P | 135 € | 216 € |
| Familiaritzar-se amb les dades | 5 | 5 | P | 135 € | 135 € |
| Estimació dificultat experiència | | | | | |
| Disseny del algoritme | 48 | 90 | D | 2880 € | 5400 € |
| Implementació | 24 | 40 | P | 648 € | 1080 € |
| Valoració | 48 | 50 | T | 1440 € | 1500 € |
| Agrupació dels problemes | | 5 | C | - | 375 € |
| Disseny del algoritme | 15 | - | D | 900 € | 0 € |
| Implementació | 10 | - | P | 270 € | 0 € |
| Valoració | 25 | - | T | 750 € | 0 € |
| Creació del recomanador | | | | | |
| Disseny del algoritme | 30 | 20 | D | 1800 € | 1200 € |
| Implementació | 15 | 8 | P | 405 € | 216 € |
| Valoració | 30 | 15 | T | 900 € | 450 € |
| Predicció d'abandonament | | | | | |
| Disseny del algoritme | - | 45 | D | - | 2700 € |
| Implementació | - | 15 | P | - | 405 € |
| Valoració | - | 30 | T | - | 900 € |
| Documentació | 40 | 65 | C | 3000 € | 4875 € |
| Total | 370 | 471 | | 18888 € | 25077 € |

Taula 7: Cost real en salaris comparat amb la previsió de la taula 4.

### 4.5.2 Costos indirectes

| Hardware | Consum (W/h) | Temps estimat (h) | Temps real (h) | Cost previst | Cost real |
|---|---|---|---|---|---|
| Portàtil Toshiba | 300 | 370 | 471 | 19,98 € | 25,43 € |
| Raspberry Pi B | 3,5 | 24h × 30d × 4m | 24h × 30d × 7m | 1,81 € | 3,17 € |
| Total | | | | 21,79 € | 28,60 € |

Taula 8: Despesa real en electricitat.

### 4.5.3 Recursos software

A més dels recursos software enumerats al apartat 4.4 aquest projecte també ha utilitzat la llibreria gratuïta LibSVM.

### 4.6 Resum

La taula següent resumeix el pressupost del projecte tenint en compte les desviacions:

| Secció | Pressupost | Cost |
|---|---|---|
| Hardware | 41,92 € | 41,92 € |
| Sotfware | 0,00 € | 0,00 € |
| Recursos Humans | 18888,00 € | 25077,00 € |
| Costos Indirectes | 21,79 € | 28,60 € |
| Total | 18951,71 € | 25147,52 € |

Taula 9: Resum de pressupost i el cost real del projecte.

## 5 Informe de sostenibilitat

En aquesta secció es fa un anàlisis de la sostenibilitat d'aquest projecte seguint les pautes definides a [12]. L'apartat 5.4 conté la matriu de puntuació de sostenibilitat.

### 5.1 Viabilitat econòmica

L'apartat 4 fa una descripció detallada del cost d'aquest projecte. En un futur aquest treball *haurà de ser integrat* a Jutge.org i això implicarà una despesa addicional.

Aquest projecte pretén assolir un cert nivell de qualitat: Es provaran diversos algoritmes i s'avaluaran els resultats. Es podria reduir el cost d'aquest projecte utilitzant només un algoritme però la qualitat es veuria afectada.

### 5.2 Viabilitat social

Aquest projecte esta orientat a Jutge.org, però hauria de ser adaptable a qualsevol altre sistema de jutge en línia.

Degut a la natura dels jutges en línia se espera que els beneficiaris del projecte siguin ciutadans del primer món amb accés a Internet i relacionats amb al desenvolupament de software.

Finalment s'utilitzarà aquesta informació per fer un recomanador de problemes. Els usuaris seran lliures d'escollir fer cas al recomanador i aquest no els hauria d'afectar negativament de cap manera.

Cap col·lectiu sortirà perjudicat d'aquest projecte.

### 5.3 Viabilitat ambiental

Aquest projecte ha consumit 158,94 kWh (taula 8) que són aproximadament uns 131,93 kg de $CO_2$[11].

Quan aquest treball sigui integrat a Jutge.org el seu cost elèctric serà menyspreable.

Quan sigui necessari millorar el treball, un nou anàlisis mediambiental serà necessari per tenir en compte els possibles avanços en energies renovables.

## 5.4 Taula de sostenibilitat

En base a les qüestions comentades als apartats anteriors, la taula següent mostra la puntuació d'aquest treball per a cada punt de vista analitzat.

| | PPP | Vida útil | Riscos | Total |
|---|---|---|---|---|
| Econòmic | 8 | 15 | -1 | 22 |
| Social | 9 | 20 | 0 | 29 |
| Ambiental | 9 | 18 | 0 | 27 |
| Total | 26 | 53 | -1 | 78 |

Taula 10: Matriu de sostenibilitat.

# 6 Descripció de les dades

En aquesta secció es descriu en detall com són les dades que disposem per fer aquest treball.

## 6.1 Descripció de la base dades

Les dades que es descriuen en aquest apartat formen part de la base de dades relacional de Jutge.org.

Disposem d'una còpia parcial *anonimitzada*, realitzada el 4 de desembre 2015, on informació personal, com el nom o la direcció de correu, ha estat eliminada; per referir-nos als usuaris només disposem de l'identificador intern que assigna Jutge.org, del qual parlarem més endavant a Users.

De tota la base de dades nomes hem agafat aquelles taules que contenien dades que podien ser rellevants per aquest treball, aquelles que tenen a veure amb els problemes o els enviaments dels usuaris.

### 6.1.1 Composició de les dades

Les taules per a les que disposem informació queden representades en la figura següent:

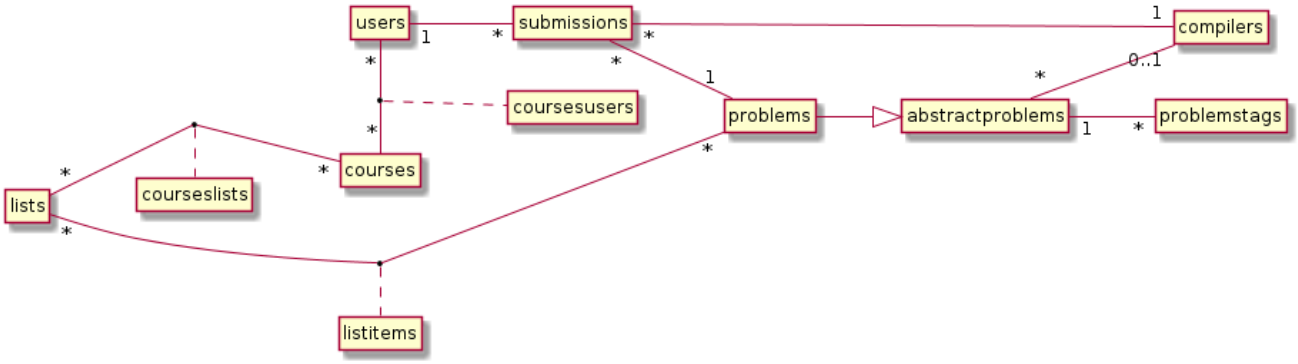


Figura 4: Diagrama UML de la base de dades mostrant les relacions més importants.

La figura anterior mostra l'esquema UML de la base de dades en el que trobem representades les relacions més importants per aquest treball. Les relacions * - * (molts - molts) no es poden implementar directament i per això apareixen les taules que estan representades amb una línia discontinua.

La figura 5 mostra com esta implementada internament la base de dades amb els camps de les taules.

A continuació es descriuran les taules i els seus camps més rellevants.

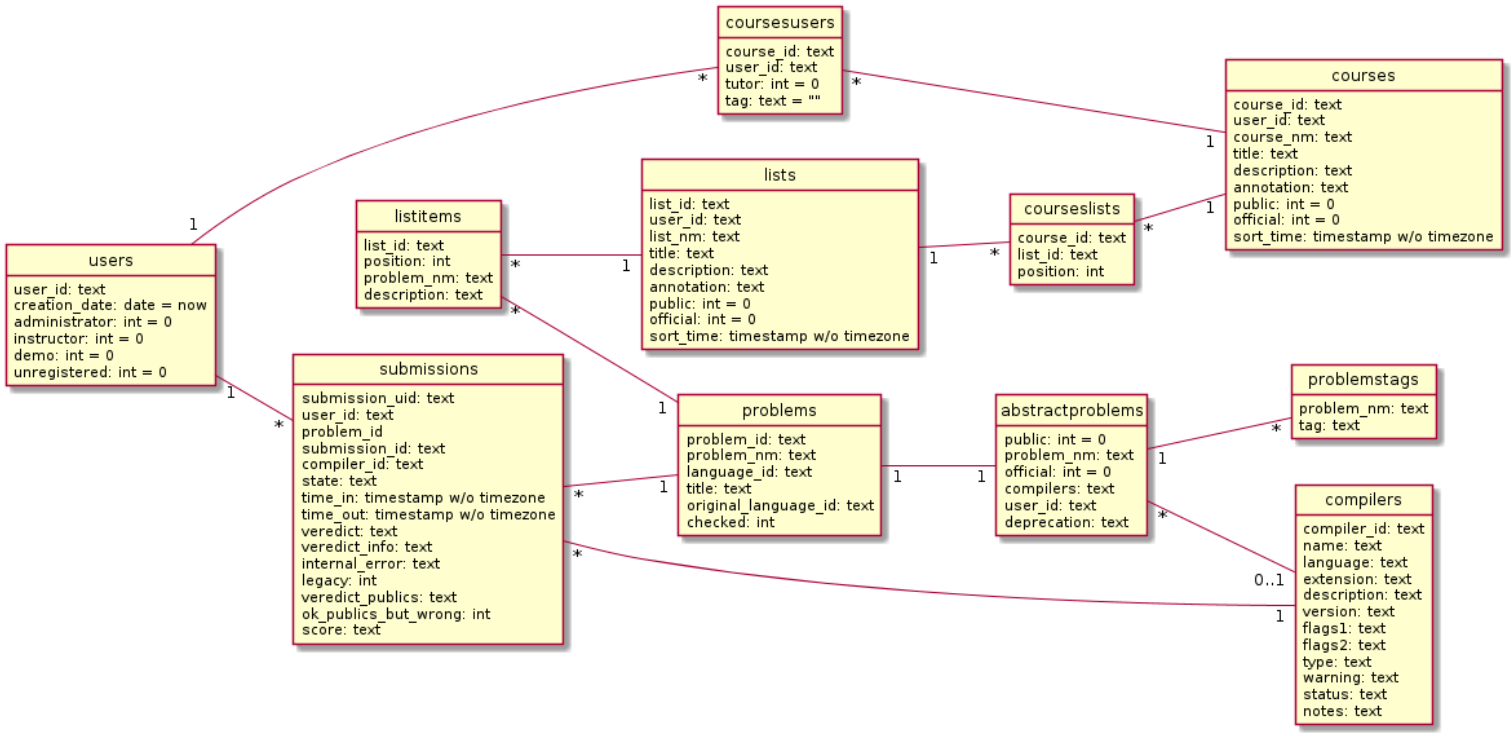


Figura 5: Diagrama UML de la implementació de la base dades mostrant només les relacions més importants.

### Users

Juntament amb les taules Problems i Submissions aquesta és una de les taules més importants. Conté informació sobre els usuaris.

- *user_id*: És tracta d'un identificador únic d'usuari; el caràcter U seguit de 5 dígits, per exemple: U12345. Aquests números es generen de forma seqüencial independentment dels usuaris i per tant no aporten informació sobre aquests.
- *administrador*, *instructor*, *demo*, *unregistered*: Son valors booleans que indiquen si l'usuari és un administrador, instructor, un compte demo o si el compte ha estat esborrat.

S'ignora l'usuari U00000 perquè és un usuari virtual que utilitza el sistema quan està realitzant comprovacions internes.

### AbstractProblems

Aquesta taula conte informació general sobre els problemes. Es tracta d'aquella informació que no depèn del idioma i que no canvia mai.

- *problem_nm*: L'identificador del problema; el caràcter P seguit de 5 dígits. Identifica inequívocament un problema; per exemple: P62627.
- *user_id*: Es tracta del identificador d'usuari de la persona que l'ha creat.
- *compilers*: En cas què el problema tingués cap restricció de compilador en aquest camp es referenciaria el nom del compilador.

Els altres camps no són rellevants per aquest treball, a més, *public* i *offical* no estan en ús.

### Problems

La taula descrita al apartat anterior, AbstractProblems, conté la informació general d'un problema, la seva part abstracta. Aquesta taula emmagatzema les traduccions dels problemes, la part específica.

Cada problema pot estar en diversos idiomes, això afecta al títol, l'enunciat, els jocs de prova i fins i tot la solució; tot i ser el mateix problema els textos poden ser diferents i això fa que una nova solució sigui necessària.

- *problem_id*: És l'únic identificador que no segueix la forma dels altres identificadors. Es tracta del valor del camp *problem_nm* seguit del caràcter "_" seguit del valor de *language_id*. D'aquesta manera només mirant aquest camp es pot saber de quin problema es tracta i en quin idioma es troba. Per exemple: P62627_es, seria l'idioma espanyol del problema P62627, mentre que P62627_en representaria la seva versió anglesa.
- *problem_nm*: És l'identificador del problema, tal i com esta descrit a AbstractProblems.
- *language_id*: Es tracta d'un identificador del idioma, actualment només: es, ca, en i fr s'utilitzen per al espanyol, català, anglès i francès respectivament.
- *checked*: És un valor booleà que indica si s'ha verificat la correctesa de la solució proposada pel creador del problema.

Els altres camps no són rellevants per a aquest treball, són el títol del problema, l'idioma original, el seu traductor ...

### Submissions

Aquesta és una de les taules més importants de la base de dades i la que més informació conté. Aquesta taula guarda cada enviament fet per cada usuari per a cada problema.

- *submission_uid*: Identificador únic d'enviament. El caràcter S seguit de 9 dígits, p.e. S000300859.
- *user_id*: L'usuari que ha fet l'enviament. És el mateix camp que es troba a descrit Users.
- *problem_id*: El problema. És l'identificador del problema amb l'idioma tal i com esta descrit a Problems.
- *submission_id*: Identifica el número d'enviament per a cada usuari i problema, començant per 1. Esta format pel caràcter S seguit de 3 dígits; p.e. S002.

  Cada usuari pot fer diferents enviaments al mateix problema per tal d'arreglar els errors o per provar noves maneres de resoldre el problema. Aquest camp serveix per identificar un intent concret a un problema.
- *compiler_id*: El compilador utilitzat que s'ha utilitzat.
- *time_in*: Moment en el que s'ha enviat.
- *time_out*: Moment en la que s'ha acabat la correcció.

  Un detall a tenir en compte és que es realitzen revaluacions de forma esporàdica i això actualitza aquest camp. Per això es possible trobar-se valors de *time_in time_out* amb més d'un any de diferència.
- *veredict*: El resultat de la correcció [13].
  - AC: Acceptat.
  - PE: Error de presentació.
  - WA: Resposta Incorrecta.
  - IC: Caràcter Invalid.
  - EE: Error d'execució.
  - CE: Error de compilació.
  - NC: La solució no segueix les limitacions del problema.
  - SE: Solució donada pel creador del problema es incorrecta.
  - IE: Error intern de Jutge.
  - FE: Error fatal.

Els altres camps formen part del procés de correcció i mostrat del error i no són importants per aquest treball.

### Compilers

Informació sobre els compiladors que suporta Jutge.org. La informació d'aquesta taula no és massa rellevant per a aquest treball. L'únic camp que ens podria interessar es *compiler_id*, és el nom del compilador.

**Lists**

Els problemes s'agrupen en llistes ordenades segons els criteri del creador de la llista.

Aquesta taula conté informació general sobre les llistes; com el seu nom o una descripció.

- *list_id*: L'identificador de la llista; caràcter L seguit de 5 dígits.
- *user_id*: El creador de la llista.
- *list_nm*: El nom de la llista en un format intern.
- *public* i *official* indiquen si es tracta d'una llista publica o oficial respectivament.

**ListItems**

Aquesta taula serveix per relacionar els problemes amb les llistes a les que pertanyen.

- *list_id*: L'identificador de la llista; igual que a Lists.
- *problem_id*: L'identificador del problema, mateix camp que a Problems.
- *position*: Posició del problema a la llista.

**Courses**

A més les llistes formen part de cursos. En poques paraules un curs és una agrupació ordenada de llistes on els usuaris s'apunten.

En aquesta taula es troba la informació general sobre els cursos. Com el seu nom, si és public o privat o qui és el seu creador.

- *course_id*: És l'identificador únic del curs; caràcter C seguit de 5 dígits, p.e. C00199.
- *user_id*: L'identificador del creador del curs.
- *course_nm*: El nom del curs com a identificador intern.
- *public*: Booleà que indica si el curs és públic o si és privat.

**CoursesLists**

Aquesta taula relaciona les llistes amb els cursos.

- *course_id*: L'identificador del curs tal i com és troba descrit al apartat Courses.
- *list_id*: L'identificador de la llista definit a Lists.
- *position*: Posició que ocupa aquesta llista respecte a les altres llistes del curs.

**CoursesUsers**

Els usuaris han de poder unir-se a diferents cursos. Aquesta taula relaciona els usuaris amb els cursos.

- *course_id*: És l'identificador del curs descrit a Courses.
- *user_id*: És l'identificador dels usuaris descrit a Users.

**ProblemsTags**

Els problemes disposen tags, són paraules que el creador del problema li ha assignat. Aquesta taula és únicament un camp de text que conté la paraula i l'identificador que la relaciona amb un problema.

Per desgràcia aquests tags no són visibles per als usuaris i la major part d'usuaris no saben que existeixen. És tracta d'una funcionalitat que mai s'ha arribat a utilitzar.

### 6.1.2 Mida de les dades

La taula 11 mostra el nombre de dades amb les que estem treballant.

| | |
|---|---:|
| Users | 10565 |
| AbstractProblems | 1909 |
| Problems | 2911 |
| Submissions | 1605270 |
| Compilers | 39 |
| Lists | 607 |
| ListItems | 7473 |
| Courses | 112 |
| CoursesLists | 868 |
| CoursesUsers | 22659 |
| ProblemsTags | 5304 |

Taula 11: Mida de les taules de la base de dades el 4 de desembre de 2015.

## 6.2 Descripció dels cursos

Amb la intenció d'ajudar-nos a avaluar els algoritmes es disposa de les notes, *totalment anonimitzades*, dels alumnes dels cursos de Programació 1 de diferents anys i les notes d'un curs de Programació de la Facultat de Matemàtiques.

Aquest cursos físics tenen la seva versió virtual a la base de dades de Jutge.org i nosaltres ens centrarem en aquest últim. Utilitzant els identificadors d'usuari dels alumnes, *user_id*, i l'identificador del curs corresponent, *course_id*, s'extrauran tots els problemes que pertanyen al curs i tots els enviaments d'aquests usuaris en el període del curs.

**Les dates d'un curs:** Definim que els cursos de tardor comencen el 10 de Setembre i acaben el 20 de Gener; els de primavera comencen el 8 de Febrer i acaben el 22 de Juny.

### PRO1 Tardor 2013 - 2014

El curs de Programació 1 de la FIB del quadrimestre de tardor 2013 - 2014. La taula següent resumeix el curs.

| PRO1-2014: Programació 1 Tardor 2013 - 2014 | |
|---|---|
| Nombre de problemes | 300 |
| Nombre d'alumnes | 414 |
| Nombre de problemes sense AC | 4 |
| Nombre d'alumnes sense AC | 3 |
| Percentatge d'abandó (Nota final 0) | 21.25 % |
| Percentatge d'aprovats | 38.16 % |
| Nota mitjana | 3.53 |

Taula 12: Descripció del curs Programació 1 Tardor 2013 - 2014.

### PRO1 Tardor 2014 - 2015

El curs de Programació 1 de la FIB del quadrimestre de tardor 2014 - 2015. La taula següent resumeix el curs.

| PRO1-2015: Programació 1 Tardor 2014 - 2015 | |
|---|---|
| Nombre de problemes | 303 |
| Nombre d'alumnes | 447 |
| Nombre de problemes sense AC | 6 |
| Nombre d'alumnes sense AC | 4 |
| Percentatge d'abandó (Nota final 0) | 24.60 % |
| Percentatge d'aprovats | 41.16 % |
| Nota mitjana | 3.80 |

Taula 13: Descripció del curs Programació 1 Tardor 2014 - 2015.

### FME Tardor 2015 - 2016

El curs de Programació de la Facultat de Matemàtiques i Estadística de tardor 2014 - 2015. La taula següent resumeix el curs.

| FME-2016: FME Tardor 2015 - 2016 | |
|---|---|
| Nombre de problemes | 218 |
| Nombre d'alumnes | 69 |
| Nombre de problemes sense AC | 16 |
| Nombre d'alumnes sense AC | 0 |
| Percentatge d'abandó (Nota final 0) | 11.59 % |
| Percentatge d'aprovats | 57.97 % |
| Nota mitjana | 5.02 |

Taula 14: Descripció del curs FME Tardor 2015 - 2016.

**PRO1 Tardor 2015 - 2016**

El curs de Programació 1 de la FIB del quadrimestre de tardor 2015 - 2016. La taula següent resumeix el curs.

| PRO1-2016: Programació 1 Tardor 2015 - 2016 | |
|---|---|
| Nombre de problemes | 313 |
| Nombre d'alumnes | 441 |
| Nombre de problemes sense AC | 8 |
| Nombre d'alumnes sense AC | 1 |
| Percentatge d'abandó (Nota final 0) | 16.78 % |
| Percentatge d'aprovats | 54.64 % |
| Nota mitjana | 4.60 |

Taula 15: Descripció del curs Programació 1 Tardor 2015 - 2016.

## 6.3 Descripció dels logs d'accés

Com a servei web, Jutge.org genera registres d'informació (logs) sobre totes les peticions que rep. Aquests logs tenen la finalitat d'ajudar als administradors a resoldre possibles problemes [23] o inclús per detectar atacs contra el servei [21, 24].

Els logs també contenen, en molts casos, un esquema casi perfecte de com un usuari interactua amb un servei i a partir d'aquí es poden extreure conclusions útils [7].

Disposem dels logs d'accés de Jutge.org des de setembre de 2013 fins a maig de 2016. La taula següent descriu els camps que tenen els logs de Jutge.org:

| Nom del camp | Descripció del camp |
|---|---|
| Data | Data exacta de la petició, p.e: `2015-11-10 00:00:02` |
| IP | Direcció IP de l'ordinador que esta fent la petició, p.e: `1.1.1.1` |
| Tipus | Tipus de petició HTTP, p.e: `GET` |
| Direcció | Direcció web a la que va dirigida la petició, p.e: `/problems/P78142_ca/submissions` |
| User_id | Identificador d'usuari que ha fet la petició. Pot ser buit si l'usuari no s'ha identificat, p.e: `U00001` |
| Sessió | Sessió de l'usuari que ha fet la petició, p.e: `sakb8fpufh3jkuu8p3nc21de2amdd4li` |
| Domini | Domini al que va dirigida la petició, p.e: `jutge.org` |

Taula 16: Descripció de les dades que conté un log de Jutge.org.

Els camps *sessió* i *domini* són camps recents que no estan als logs de 2013. *sessió* va ser introduït al Octubre de 2014 i *domini* al març de 2015.

## 6.4 Detalls sobre la confidencialitat de les dades

En aquest apartat es resumeixen totes les qüestions de confidencialitat esmentades anteriorment.

- La base de dades ens ha estat donada anonimitzada. L'única informació identificativa dels usuaris és l'identificador: ***user_id***.
- Es disposa de les notes d'alguns estudiants per alguns cursos. Ens han estat donades anonimitzades: els noms dels estudiants han estat reemplaçats pels seus identificadors d'usuari, ***user_id***, p.e. `U00000 10`.
- Els logs d'accés de Jutge.org registren tota l'activitat, però l'únic camp que pot identificar un usuari és: ***user_id***.

Els identificadors d'usuari, *user_id*, es generen de forma seqüencial: van començar per 00001 al 2006, quan Jutge.org va entrar en funcionament, i ha estat incrementant-se des de aleshores. Per tant no aporten informació sobre els usuaris.

Per aquestes raons totes les dades que es disposen es poden considerar anònimes.

# 7 Anàlisi de la dificultat dels problemes i l'experiència dels usuaris

## 7.1 Descripció del problema

Jutge.org té molts problemes i molts usuaris, però no tots els problemes tenen la mateixa dificultat ni tots els usuaris la mateixa experiència programant.

És evident que la dificultat d'un problema depèn de l'habilitat de la persona que el fa. Usuaris més experimentats tendeixen a considerar els problemes com a més fàcils, mentre que usuaris amb menys experiència els veuen més difícils.

Per altra banda, l'experiència dels usuaris augmenta amb el temps. A mesura que els usuaris van realitzant problemes, van aprenent i la seva experiència augmenta. Això fa que l'experiència sigui un concepte dinàmic.

A més, l'experiència d'una persona és un concepte difícil de mesurar empíricament; en els entorns acadèmics es realitzen exàmens i proves per mesurar-ho. Nosaltres no disposem de la capacitat de posar a prova als usuaris directament, però sí que disposem de la informació dels problemes que han resolt o no. Farem servir els problemes que han fet com a mesura de la seva habilitat.

Només ens centrarem en aquells problemes que han completat, aquells que tenen un *veredicte AC*.

Volem establir puntuacions als problemes i usuaris de manera que aquestes puguin ser utilitzades empíricament com a indicadors de la dificultat i l'experiència. Per aconseguir-ho provarem tres algoritmes diferents que descriurem en detall a continuació i més tard valorarem experimentalment la seva eficàcia.

## 7.2 Algoritmes

De tota la informació que disposem, ens centrarem només en si un usuari a resolt un problema o no (*veredicte AC*). Per descriure l'estat ens valem conceptualment d'una matriu booleana $\boldsymbol{M}$ de mida $P \times U$ on $P$ i $U$ són el nombre de problemes i d'usuaris respectivament.

$$\boldsymbol{M} = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1U} \\ m_{21} & m_{22} & \dots & m_{2U} \\ \vdots & \vdots & \ddots & \vdots \\ m_{P1} & m_{P2} & \dots & m_{PU} \end{pmatrix}$$

on

$$m_{ij} = \begin{cases} 1 & \text{si el problema } i \text{ ha estat resolt per l'usuari } j \\ 0 & \text{en cas contrari} \end{cases}$$

Hem ideat tres algoritmes diferents que introduirem a continuació. A mesura que presentem un nou algoritme mostrarem el seu funcionament amb un petit

exemple de 6 problemes i 10 usuaris que queda representat com a:

$$\boldsymbol{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Per comoditat, anomenarem als usuaris per número de columna: $u_1$, $u_2$, ..., $u_9$, $u_{10}$. Anàlogament per als problemes: $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$. *Sobrecarregarem aquesta notació i utilitzarem també $\boldsymbol{p}_i$ i $\boldsymbol{u}_j$ per referir-nos a les i files, j columnes de la matriu $\boldsymbol{M}$ respectivament.*

Els diferents algoritmes retornen resultats de formes diferents, per poder fer una comparació entre ells, calcularem el rànquing ordinal dels resultats i després calcularem la *distància Kendall Tau* [14] que ve donada per:

Siguin $L_1$ i $L_2$ dues llistes qualsevol:

$$K(\tau_1, \tau_2) = |\{(i,j) : i < j, (\tau_1(i) < \tau_1(j) \wedge \tau_2(i) > \tau_2(j)) \\ \vee (\tau_1(i) > \tau_1(j) \wedge \tau_2(i) < \tau_2(j))\}| \quad (1)$$

On $\tau_1(i)$, $\tau_2(i)$ és el rànquing de $i$ a les llistes $L_1$ i $L_2$ respectivament.

D'aquesta manera obtindrem un valor numèric que ens indicarà com de diferents són dos resultats. Com que els valors que retorna (1) depénen de la mida de les llistes, els normalitzarem dividint per $n(n-1)/2$, d'aquesta manera aconseguirem valors en l'interval $[0, 1]$.

### 7.2.1 Algoritme Naïf

Aquest algoritme aplica un criteri molt senzill que es pot resumir en dues frases: Els problemes més difícils són els que menys usuaris han aconseguit, i els "millors" usuaris són els que han resolt més problemes.

En aquest algoritme, trobar la dificultat consisteix únicament en sumar el nombre d'usuaris que l'han resolt. L'equació (2) descriu el procés en forma d'operacions vectorials. És important tenir en compte que els valors més grans representen els problemes més fàcils.

$$\boldsymbol{d} = \boldsymbol{M} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad (2)$$

Anàlogament calcular l'experiència dels usuaris és igual de senzill, només cal sumar els problemes resolts. Però aquest cop els valors alts indiquen experiència.

$$\boldsymbol{e} = \boldsymbol{M}^{\boldsymbol{T}} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \quad (3)$$

Aplicant (2) al nostre exemple:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 3 \\ 6 \\ 4 \\ 3 \end{bmatrix}$$

Si ordenem el vector de forma creixent podem veure quins són els problemes més difícils: $p_2 \geq p_3 \geq p_6 \geq p_5 \geq p_1 \geq p_4$. El $p_2$ només l'han resolt 2 usuaris, per tant per a nosaltres és el problema més difícil; mentre que el problema $p_4$ ha estat resolt per 6 usuaris i això el fa el més fàcil.

Per a l'habilitat dels usuaris, apliquem (3) i calculem de la mateixa manera:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}^T \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 3 \\ 1 \\ 0 \\ 3 \\ 2 \\ 2 \\ 3 \\ 5 \end{bmatrix}$$

Per tant, $u_5 \leq u_4 \leq u_1 \leq u_7 \leq u_8 \leq u_2 \leq u_3 \leq u_6 \leq u_9 \leq u_{10}$. L'usuari $u_5$ no ha resolt cap problema, per tant és l'usuari amb menys experiència; l'usuari $u_{10}$ els ha resolt gairebé tots així que el considerem l'usuari més experimentat.

Com podem veure, aquest algoritme és molt senzill però pot servir com a base per comparar altres algoritmes.

### 7.2.2 JutgeRank

El nostre segon algoritme treu inspiració del famós algoritme de Google:

**PageRank** PageRank és un rànquing per a les pàgines web basat en la seva importància.

Intuïtivament una pàgina web ha de ser important (sense tenir en compte el contingut) si la referencien moltes pàgines web. Però no totes les referències tenen el mateix significat, una referència des de una pàgina important és molt més rellevant que una des de una pàgina poc important [20].

PageRank entén la web com un graf dirigit on les pàgines són nodes i els enllaços entre les pàgines són els arcs entre nodes. A partir d'aquí cada pàgina aporta puntuació a les pàgines cap a les que té sortida i rep puntuació de les pàgines entrants.

**JutgeRank** Nosaltres podem entendre les dades de Jutge.org com un graf bipartit de problemes a usuaris on $\boldsymbol{M}$ és la seva matriu d'adjacència. La figura 6 representa el nostre exemple interpretat com a gràfic bipartit.

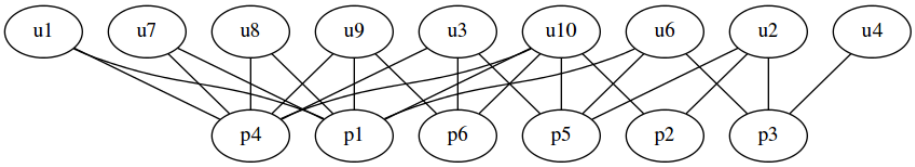


Figura 6: Representació com a graf bipartit de la matriu d'exemple descrita al apartat 7.2. L'usuari 5 (u5) no apareix perquè no ha realitzat cap problema; seria un node sense cap connexió.

El concepte és que cada usuari aporta dificultat als problemes i cada problema aporta experiència als usuaris.

JutgeRank defineix la dificultat d'un problema com la mitjana de l'experiència dels usuaris que l'han resolt:

$$\boldsymbol{d}_i = \begin{cases} 1 & \text{si ningú ha fet el problema } i \\ \dfrac{\sum p_i}{|p_i|} & \text{en cas contrari} \end{cases} \tag{4}$$

on

$$p_i = \{\boldsymbol{e}_j | \boldsymbol{M}_{ij} = 1\}$$

I, calcula l'experiència dels usuaris com el mòdul de la dificultat dels problemes han resolt:

$$\boldsymbol{e}_j = ||(\boldsymbol{d}_1 \cdot \boldsymbol{M}_{1j}, \boldsymbol{d}_2 \cdot \boldsymbol{M}_{2j}, \ldots, \boldsymbol{d}_P \cdot \boldsymbol{M}_{Pj})|| \tag{5}$$

Per tal que els valors és mantinguin controlables, el vector dificultat és normalitza a cada iteració convertint-lo en un vector unitari, és a dir: $||\boldsymbol{d}|| = 1$. Amb aquesta definició recursiva podem anar calculant la dificultat i l'experiència dels usuaris.

El procés acaba quan per algun petit $\epsilon$ és compleix:

$$|\boldsymbol{d}^{anterior} - \boldsymbol{d}^{actual}| < \epsilon$$

i s'assumeix convergència.

Inicialment establim la dificultat dels problemes com a 1, tots són difícils. L'algoritme de JutgeRank queda resumit en aquests dos passos:

1. Establir inicialment $\boldsymbol{d} = 1$ i aplicar (5) per trobar $\boldsymbol{e}$ inicial.
2. Aplicar (4) i (5) fins convergència.

En el nostre exemple:

$$\boldsymbol{d} = \begin{bmatrix} 0.37 \\ 0.48 \\ 0.32 \\ 0.38 \\ 0.44 \\ 0.45 \end{bmatrix} \text{ i } \boldsymbol{e} = \begin{bmatrix} 0.53 \\ 0.72 \\ 0.73 \\ 0.32 \\ 0.0 \\ 0.66 \\ 0.53 \\ 0.53 \\ 0.69 \\ 0.95 \end{bmatrix}$$

Si ordenem els vectors per veure una relació similar a la que hem vist al subapartat 7.2.1, obtenim per als problemes: $p_3 \leq p_1 \leq p_4 \leq p_5 \leq p_6 \leq p_2$. Es tracta d'un ordre diferent al que havíem obtingut però no tant diferent: només $p_3$ està en un altre lloc; que $p_1$ i $p_4$ s'hagin girat no és massa significatiu. La distància Kendall Tau, (1), entre aquest resultat i el donat per l'algoritme Naïf és de 0.33.

Per als usuaris: $u_5 \leq u_4 \leq u_1 \leq u_7 \leq u_8 \leq u_6 \leq u_9 \leq u_2 \leq u_3 \leq u_{10}$. Només la parella $u_2$,$u_3$ s'ha canviat amb $u_6$,$u_9$ i ens dona una distància Kendall Tau de 0.09.

### 7.2.3 SVD

El tercer algoritme es basa en la Singular Value Descomposition (SVD). Segons el teorema de la descomposició en valors singulars [3] tota matriu $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ es pot descompondre en el producte de dos matrius ortonormals, $\boldsymbol{U} \in \mathbb{R}^{m \times m}$, $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, i una matriu pseudo-diagonal $\boldsymbol{D} = diag(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$ amb $p = \min(m, n)$ (és a dir, tots els element excepte els $p$ primers de la diagonal són 0), de forma:

$$\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{D}\,\boldsymbol{V}^{\boldsymbol{T}} \tag{6}$$

Depenent dels valors de $m$ i $n$ la descomposició denotada per (6) s'acostuma a donar d'una forma més compacta; en particular, si $m > n$ llavor les últimes $m - n$ columnes de $\boldsymbol{U}$ i les últimes $m - n$ files de $\boldsymbol{D}$ és poden ometre.

Si denotem amb $\boldsymbol{u_i}$ i $\boldsymbol{v_i}$ les columnes de $\boldsymbol{U}$ i $\boldsymbol{V}$ respectivament, llavors podem reescriure (6) com la suma ponderada dels seus productes:

$$\boldsymbol{A} = \sum_{i=1}^{p} \sigma_i \boldsymbol{u_i} \boldsymbol{v_i^T} \tag{7}$$

Per conveni, els elements diagonals $\sigma_i$ de $\boldsymbol{D}$, que se'ls anomena valors singulars, són no negatius i estan ordenats de forma decreixent, és a dir, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p$. Mirant (7) podem veure que $\sigma_i$ denota els pesos de les columnes de $\boldsymbol{U}$ i $\boldsymbol{V}$ i el fet que s'ordeni de forma decreixent fa que les primeres columnes de $\boldsymbol{U}$ i $\boldsymbol{V}$ siguin les més importants. Aquesta ordenació és pot explotar en forma

d'aproximació a la matriu; agafant un nombre $k < p$ de columnes:

$$\boldsymbol{A} \approx \sum_{i=1}^{k} \sigma_i \boldsymbol{u_i} \boldsymbol{v_i^T} \tag{8}$$

Aquesta aproximació s'anomena "$k$-mode truncation" (o "$k$-mode projection") de la matriu $\boldsymbol{A}$. Segons el teorema (7), (8) és una bona aproximació si els valors singulars decauen suficientment ràpid a mesura que s'incrementa $i$. La figura 7 mostra l'efecte de la reconstrucció (8) aplicat a una imatge.

En els nostres experiments no és veu que $k$ afecti massa a la precisió d'aquest problema i per tant escollirem $k = 1$.

Aplicant SVD podem descompondre la matriu $\boldsymbol{M}$ del problema i treballar amb les seves $\boldsymbol{U}$, $\boldsymbol{D}$, $\boldsymbol{V}$. Per (6) la nostra matriu $\boldsymbol{M}$ queda com:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \boldsymbol{U}\,\boldsymbol{D}\,\boldsymbol{V}^{\boldsymbol{T}}$$

$$\boldsymbol{U} = \begin{pmatrix} 0.58 & 0.22 & 0.55 & -0.17 & 0.34 & -0.41 \\ 0.20 & -0.36 & -0.19 & -0.80 & -0.36 & -0.14 \\ 0.13 & -0.60 & 0.48 & 0.40 & -0.48 & -0.04 \\ 0.60 & 0.37 & -0.08 & 0.06 & -0.38 & 0.60 \\ 0.36 & -0.57 & -0.25 & 0.07 & 0.60 & 0.33 \\ 0.35 & 0.03 & -0.60 & 0.39 & -0.16 & -0.58 \end{pmatrix}$$

$$\boldsymbol{D} = \begin{pmatrix} 3.76 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 2.27 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.65 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.98 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.74 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.68 \end{pmatrix}$$

$$\boldsymbol{V} = \begin{pmatrix} 0.31 & 0.26 & 0.28 & -0.11 & -0.05 & 0.27 \\ 0.18 & -0.67 & 0.03 & -0.33 & -0.33 & 0.21 \\ 0.35 & -0.08 & -0.56 & 0.54 & 0.09 & 0.51 \\ 0.04 & -0.26 & 0.29 & 0.41 & -0.65 & -0.06 \\ 0.00 & 0.00 & 0.00 & 0.00 & -0.00 & -0.00 \\ 0.29 & -0.42 & 0.48 & 0.32 & 0.62 & -0.18 \\ 0.31 & 0.26 & 0.28 & -0.11 & -0.05 & 0.27 \\ 0.31 & 0.26 & 0.28 & -0.11 & -0.05 & 0.27 \\ 0.40 & 0.27 & -0.08 & 0.29 & -0.27 & -0.59 \\ 0.55 & -0.14 & -0.35 & -0.45 & 0.06 & -0.31 \end{pmatrix}$$

Amb la matriu $\boldsymbol{M}$ descomposta s'han ideat dues tècniques per extreure els resultats desitjats:

**SVD1**

La primera columna de $\boldsymbol{U}$ i de $\boldsymbol{V}$ són les columnes més importants. Definim la dificultat com la primera columna de $\boldsymbol{U}$ i definim l'experiència com la primera columna de $\boldsymbol{V}$.

Si es calcula la dificultat s'obté: $p_3 \geq p_2 \geq p_6 \geq p_5 \geq p_1 \geq p_4$; un orde que no és especialment diferent. La distància Kendall Tau amb el rànquing Naïf és: 0.07.

Per a l'experiència s'obté: $u_5 \leq u_4 \leq u_2 \leq u_6 \leq u_7 \leq u_8 \leq u_1 \leq u_3 \leq u_9 \leq u_{10}$. La distància Kendall Tau amb el rànquing Naïf és: 0.2.

**SVD2**

Aquesta tècnica s'inspira en l'ús de la SVD per a la compressió d'imatges, descrit la figura 7.

Segons l'equació (8) podem reconstruir una matriu "sense soroll" on només estiguin present les dades més significatives. Un cop tinguem aquesta aproximació de $\boldsymbol{M}$ "sense soroll" aplicarem (2) i (3) sobre ella.

La reconstrucció parcial de la nostra matriu d'exemple:

$$\begin{pmatrix} 0.68 & 0.4 & 0.76 & 0.08 & 0.0 & 0.62 & 0.68 & 0.68 & 0.88 & 1.21 \\ 0.23 & 0.14 & 0.26 & 0.03 & 0.0 & 0.21 & 0.23 & 0.23 & 0.30 & 0.41 \\ 0.16 & 0.09 & 0.18 & 0.02 & 0.0 & 0.15 & 0.16 & 0.16 & 0.21 & 0.28 \\ 0.70 & 0.41 & 0.78 & 0.08 & 0.0 & 0.64 & 0.70 & 0.70 & 0.91 & 1.24 \\ 0.43 & 0.25 & 0.48 & 0.05 & 0.0 & 0.39 & 0.43 & 0.43 & 0.55 & 0.76 \\ 0.41 & 0.24 & 0.45 & 0.05 & 0.0 & 0.37 & 0.41 & 0.41 & 0.53 & 0.72 \end{pmatrix}$$

Després d'aplicar (2) i calcular la seva distància Kendall Tau respecte Naïf obtenim: $p_3 \geq p_2 \geq p_6 \geq p_5 \geq p_1 \geq p_4$ i 0.07. Per a l'experiència: $u_5 \leq u_4 \leq u_2 \leq u_6 \leq u_7 \leq u_8 \leq u_1 \leq u_3 \leq u_9 \leq u_{10}$ i 0.2.

Ens dona la mateixa ordenació que SVD1 però pensem que es degut a la simplicitat del exemple, i que no passarà en casos reals.

## 7.3 Validació

Per valorar l'eficàcia dels algoritmes calculem l'experiència dels usuaris per a les dades que tenim. Suposem que la nota dels alumnes és un indicador de la seva experiència i calculem la distància Kendall Tau entre els resultats dels algoritmes i la nota. La taula següent mostra les distàncies Kendall Tau obtingudes:

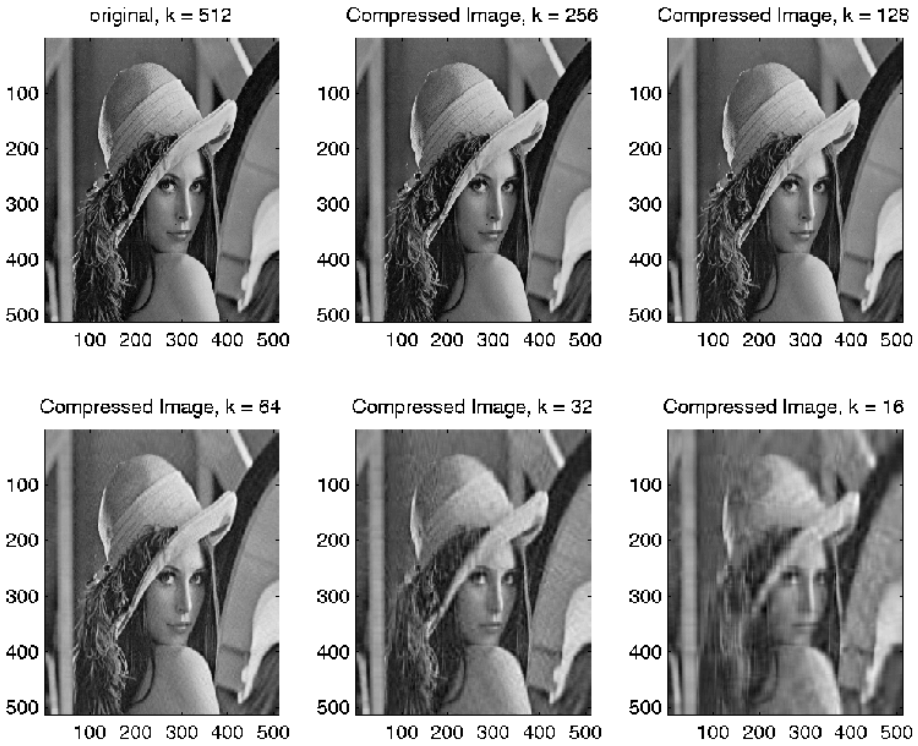


Figura 7: SVD aplicada a la compressió d'imatges. Es mostra a "lena" (512×512, escala de grisos). Comparant l'original (primera imatge) amb reconstruccions d'aquesta aplicant (8) amb diferents valors de $k$. Es pot observar com la qualitat de la imatge (exactitud de la aproximació (8)) decreix a mesura que $k$ decreix.

| | Naïf | JutgeRank | SVD1 | SVD2 |
|---|---|---|---|---|
| PRO1-2016 | 0.27 | 0.28 | 0.73 | 0.27 |
| FME-2016 | 0.36 | 0.36 | 0.62 | 0.38 |
| PRO1-2015 | 0.31 | 0.32 | 0.69 | 0.31 |
| PRO1-2014 | 0.29 | 0.30 | 0.71 | 0.29 |

Taula 17: Distància Kendall Tau normalitzada dels diferents mètodes per als respectius cursos. SVD1 té ordre invers però mateixa distància que SVD2.

Aquests valors mostren que no hi ha diferències significatives entre els algoritmes a l'hora de calcular l'experiència dels usuaris.

Les gràfiques següents mostren la *distribució* d'experiència dels usuaris i dificultat dels problemes que assignen els algoritmes a cada curs. Els valors dels algoritmes han estat normalitzats i ordenats d'usuaris de menys experiència a més, i de problemes més difícils a més fàcils.

Es pot veure com SVD2 i Naïf assignen els valors de forma molts similar. I que SVD1 és el reflexe de SVD2, és a dir, que assigna els valors de forma inversa a SVD2.

JutgeRank assigna l'experiència dels usuaris de forma molt similar, però destaca en el calcul de dificultat dels problemes. A diferència dels altres algoritmes, JutgeRank entén que els problemes difícils són aquells amb valors alts. Tot i així la seva forma d'assignar dificultats resulta més dispersa, és a dir, els altres algoritmes assignen dificultats amb valors similars.

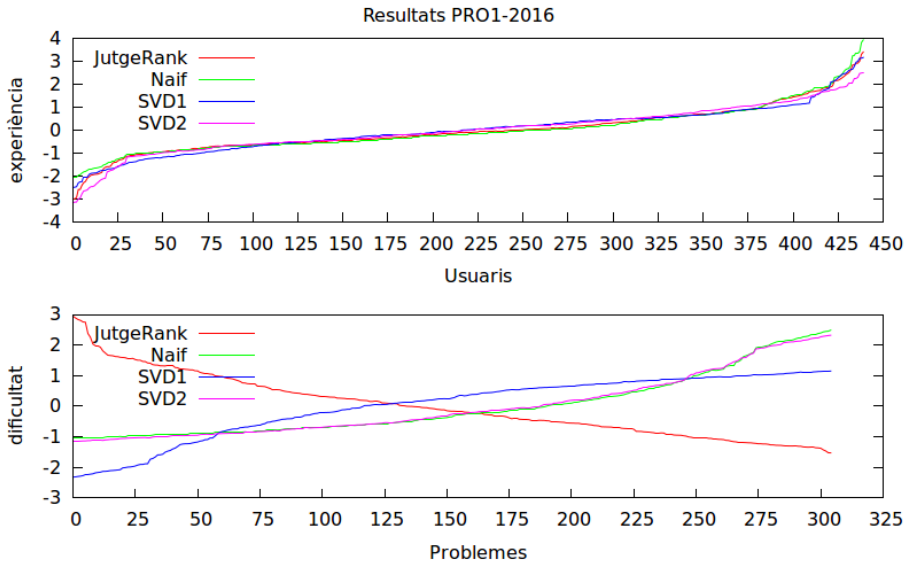


Figura 8: Proporció d'experiència dels usuaris i dificultat dels problemes del curs PRO1-2016. La gràfica ha estat ordenada de menys experiència a més i de problemes difícils a fàcils. JutgeRank defineix els problemes difícils com aquells amb valors alts.

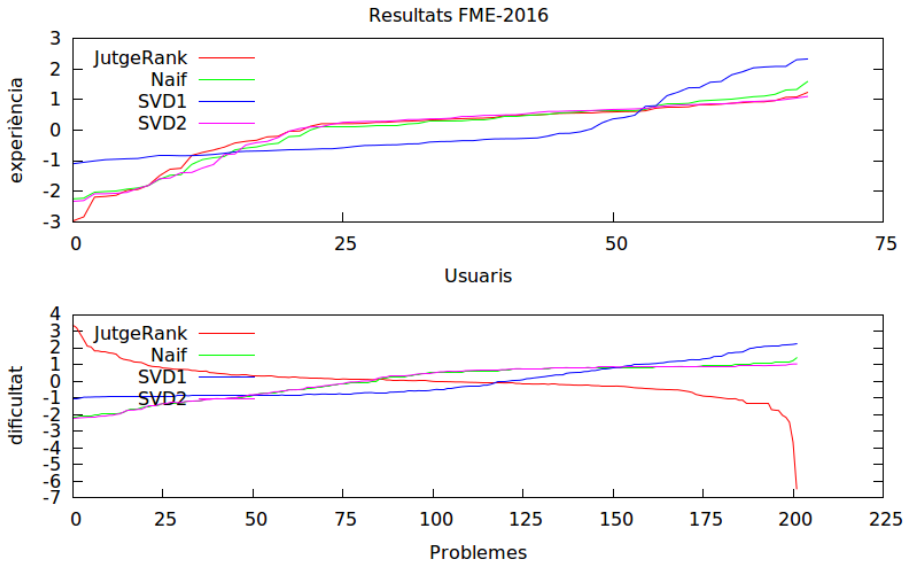


Figura 9: Proporció d'experiència dels usuaris i dificultat dels problemes del curs FME-2016. La gràfica ha estat ordenada de menys experiència a més i de problemes difícils a fàcils. JutgeRank defineix els problemes difícils com aquells amb valors alts.

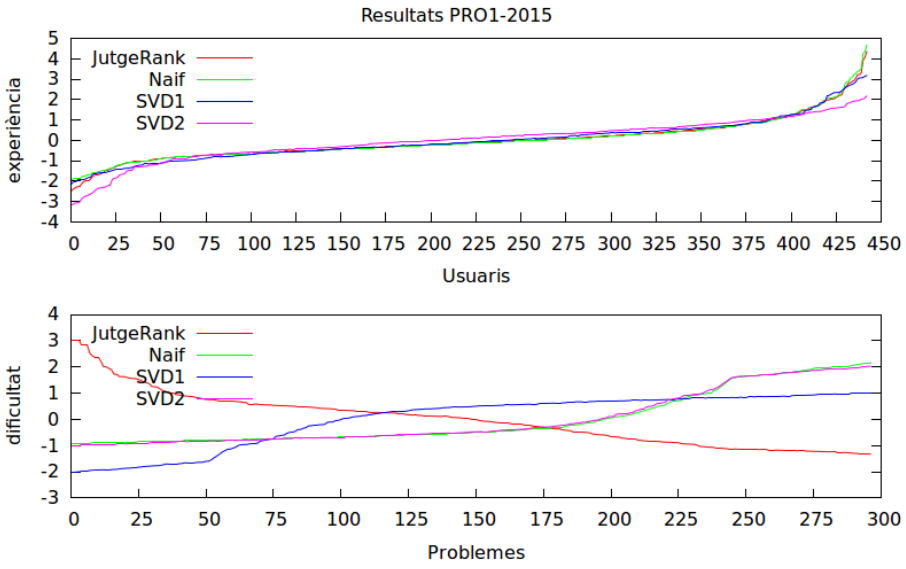


Figura 10: Proporció d'experiència dels usuaris i dificultat dels problemes del curs PRO1-2015. La gràfica ha estat ordenada de menys experiència a més i de problemes difícils a fàcils. JutgeRank defineix els problemes difícils com aquells amb valors alts.

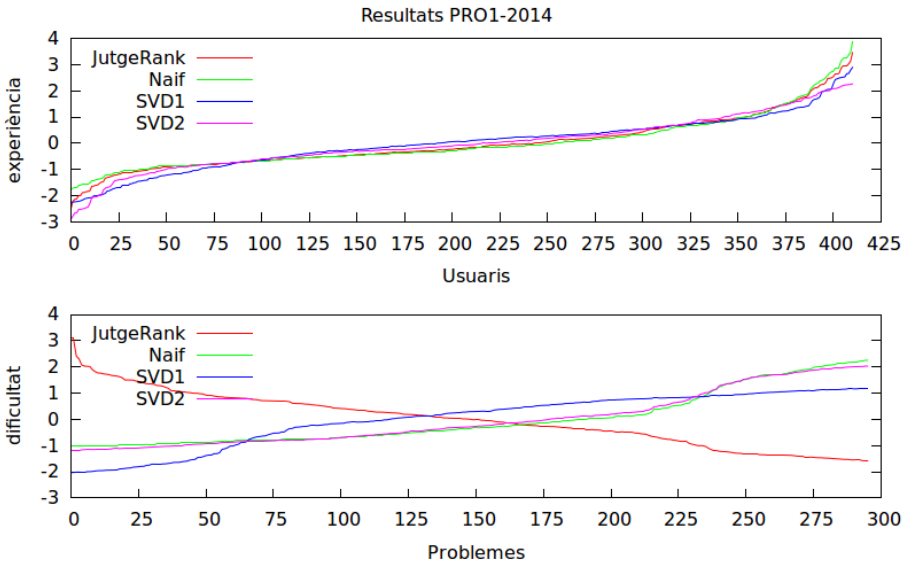


Figura 11: Proporció d'experiència dels usuaris i dificultat dels problemes del curs PRO1-2014. La gràfica ha estat ordenada de menys experiència a més i de problemes difícils a fàcils. JutgeRank defineix els problemes difícils com aquells amb valors alts.

# 8 Predicció d'abandonament i nota final dels estudiants

## 8.1 Descripció del problema

Les institucions acadèmiques estan interessades en detectar l'abandonament de les assignatures per part dels estudiants. Amb aquest coneixement es podrien prendre mesures per prevenir-lo.

Són moltes les raons que poden portar a un estudiant a abandonar. Per això la seva detecció i prevenció són tasques complexes. A més en aquest treball no es disposa *explícitament* d'informació sobre els estudiants. Només disposem de la nota final.

Però sí que es disposa de dades recollides d'una eina que utilitzen els estudiants: Jutge.org. A [16] utilitzen els logs d'un curs massiu en línia (MOOC) per determinar quan un estudiant abandona. Nosaltres disposem dels logs d'accés de Jutge.org i podem utilitzar-los de manera similar.

Un altre cosa interessant és calcular el rendiment dels estudiants. Això pot servir per prevenir a un estudiant que no està arribant al nivell necessari o per recomanar especialitats basades en el rendiment previ que ha mostrat l'estudiant.

En aquest treball també ens interessa comprovar si es pot modificar el detector d'abandonament per calcular el rendiment dels estudiants. Per calcular-ho hem decidit predir la nota final del estudiant, perquè considerem que és un bon indicador del seu rendiment.

## 8.2 Descripció de la informació extreta

En aquest treball es disposa dels logs d'accés de Jutge.org que es troben descrits a l'apartat 6.3.

Per a cadascuna de les 20 setmanes de curs (descrites a l'apartat 6.2) hem extret una sèrie de característiques detallades a la taula 18. Les característiques amb id 1–12 i 24–25 poden ser representades per un número natural i aquelles amb id 13–22 per un número real.

Després d'analitzar les característiques extretes, hem decidit descartar les setmanes 19 i 20 perquè es tracta de les últimes setmanes i l'activitat a Jutge.org és molt baixa. Finalment, totes les característiques han estat normalitzades a la unitat tipificada.

Hem utilitzat aquestes característiques per entrenar un Support Vector Machine (SVM) utilitzant la llibreria gratuïta LIBSVM[1] que ha estat portada a Julia.

Un SVM és un algoritme d'aprenentatge automàtic que determina a quina classe (agrupació) corresponen unes certes dades d'entrada.

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

El SVM entén les dades com punts multidimensionals on cada característica és una dimensió. Utilitzant una funció especial, anomenada funció kernel, és capaç de classificar els punts en les agrupacions corresponents.

**Problemes amb els logs** A l'apartat 6.3 s'ha comentat que alguns camps no es troben en els logs més antics. Per solucionar-ho hem decidit afegir les dades que falten. Per exemple, hem afegit una sessió "fantasma" a tots els usuaris que es connectaven a Jutge.org en la època en la que les sessions encara no s'enregistraven.

| ID | Característica |
|---|---|
| 1 | **Nombre de peticions**: Nombre de peticions de qualsevol tipus |
| 2 | **Nombre de sessions**: Nombre de sessions |
| 3 | **Nombre de problemes vistos**: Nombre de problemes vistos. Una petició de tipus GET a un problema |
| 4 | **Nombre d'enviaments**: Nombre d'enviaments |
| 5 | **Nombre d'enviaments consultats**: Nombre d'enviaments previs consultats. Just després de fer un enviament se fa una consulta a aquest enviament de forma automàtica per això aquest camp és sempre $\geq$ que el nombre d'enviaments |
| 6 | **Nombre de AC**: Dels enviaments realitzats, nombre de veredictes AC |
| 7 | **Nombre d'enviaments dolents**: Dels enviaments realitzats, nombre de veredictes WA + EE + CE + PE. Això representa tots els enviaments que l'estudiant no ha realitzat correctament. |
| 8 | **Nombre de WA**: Dels enviaments realitzats, nombre de veredictes WA |
| 9 | **Nombre de EE**: Dels enviaments realitzats, nombre de veredictes EE |
| 10 | **Nombre de CE**: Dels enviaments realitzats, nombre de veredictes CE |
| 11 | **Nombre de PE**: Dels enviaments realitzats, nombre de veredictes PE |
| 12 | **Altres veredictes**: Dels enviaments realitzats, nombre de veredictes que no són AC,WA,EE,CE,PE. Els altres veredictes corresponen principalment a problemes de Jutge.org. No pensem que tinguin rellevància però hem decidit mantenir el camp |
| 13 | **Problemes per sessió**: Mitjana del nombre de problemes per sessió |
| 14 | **Enviaments per sessió**: Mitjana del nombre de enviaments per sessió |
| 15 | **Problemes per dia**: Mitjana del nombre de problemes per dia de la setmana |
| 16 | **Enviaments per dia**: Mitjana de enviaments per dia de la setmana |
| 17 | **Sessions per dia**: Mitjana de sessions per dia de la setmana |
| 18 | **AC per dia**: Mitjana del nombre de veredictes AC per dia de la setmana |
| 19 | **Enviaments dolents per dia**: Mitjana del nombre de veredictes WA,EE,PE,CE per dia de la setmana |
| 20 | **WA per dia**: Mitjana del nombre de veredictes WA per dia de la setmana |
| 21 | **EE per dia**: Mitjana del nombre de veredictes EE per dia de la setmana |
| 22 | **CE per dia**: Mitjana del nombre de veredictes CE per dia de la setmana |
| 23 | **PE per dia**: Mitjana del nombre de veredictes PE per dia de la setmana |
| 24 | **Actiu1**: $\begin{cases} 1 & \text{alguna sessió aquesta setmana} \\ 0 & \text{en cas contrari} \end{cases}$ |
| 25 | **Actiu2**: $\begin{cases} 1 & \text{algun enviament aquesta setmana} \\ 0 & \text{en cas contrari} \end{cases}$ |

Taula 18: Llista de característiques extretes dels logs per a cada setmana i usuari.

## 8.3 Predicció d'abandonament

Per cadascun dels $U$ usuaris determinem si l'usuari ha deixat l'assignatura o no. Definim que un usuari ha deixat la assignatura si té una nota final de 0 o NP.

Per cadascuna de les $s$ setmanes de curs definim la matriu de dades com $\boldsymbol{X}_s^{preliminar} \in \mathbb{R}^{U\times 25}$, les files i columnes corresponen als usuaris i a les variables respectivament. Seguint com a exemple [16] hem extès la matriu de dades amb "l'històric" de les dades. Afegint a cada matriu les columnes de les prèvies matrius fins obtenir $\boldsymbol{X}_s \in \mathbb{R}^{U\times 25s}$. Podem escriure $\boldsymbol{x}^i = (x_1^i, \ldots, x_U^i)$ on $\boldsymbol{x}^i$ és la característica $i$ i $x_j^i$ és l'usuari $j$ de la característica $i$.

Hem calculat la seva puntuació Fisher [8]:

$$F(\boldsymbol{x}^j) = \frac{n_+(\mu_+^j - \mu^j)^2 + n_-(\mu_-^j - \mu^j)^2}{n_+(\sigma_+^j)^2 + n_-(\sigma_-^j)^2} \tag{9}$$

On $\mu_+^j, \sigma_+^j, \mu_-^j, \sigma_-^j$ són la mitjana i la desviació estàndard d'haver deixat l'assignatura o no respectivament, i $\mu^j$ és la mitjana de la característica. Les puntuacions obtingudes apareixen a la figura 12.

La puntuació Fisher valora com influexien els valors d'una característica sobre un altre, en aquest cas: si ha abandonat o no. Puntuacions de Fisher altes indiquen una gran rellevància, les puntuacions baixes indiquen que és poc significatiu.

A la figura 12 hem observat que la característica d'altres veredictes (id 12) obtenia valors pròxims a 0 i per això l'hem descartat. També hem notat que les característiques obtenen millors valors cap a les setmanes 9–12, això ens indica que és el millor moment per determinar si un estudiant ha abandonat.

Tenir al menys un enviament, Actiu2 (id 25), és més rellevant que tenir només una sessió, Actiu1 (id 24), durant la fase inicial del curs. Això ens indica que els estudiants que comencen esforçant-se són menys propensos a abandonar.

Al conservar l'històric apareixen moltes característiques, ens preocupa que això afecti al nostre hardware i per això hem decidit reduir dràsticament el nombre de característiques utilitzades. Hem seleccionat les 5 característiques que obtenien les millors puntuacions de Fisher: Actiu1, Actiu2, Nombre de sessions, Sessions per dia i Nombre de peticions. Finalment, i perquè al conservar l'històric apareixen moltes característiques, hem optat per seleccionar només les característiques: Actiu1, Actiu2, Nombre de sessions, Sessions per dia i Nombre de peticions.

Seguint les recomanacions dels autors de LIBSVM, hem utilitzat un kernel RBF: $K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma||\boldsymbol{x}-\boldsymbol{y}||^2}$ [5].

També hem ajustat els paràmetres $C$ i $\gamma$ amb el curs PRO1-2016. Hem utilitzat les tècniques de "grid-tunning" 5-fold cross-validation per provar totes les combinacions de $C$ i $\gamma$ i escollir aquella que obtenia millors resultats. Hem explorat els valors de $C$ entre $2^{-5}$ i $2^{12}$ i els valors de $\gamma$ entre $2^{-15}$ i $2^3$. Finalment hem obtingut els valors $C = 128$ i $\gamma = 2^{-13}$.

Hem entrenat el SVM amb el curs PRO1-2016 i els paràmetres anteriors. Després, hem calculat el ràtio de predicció amb els altres cursos per a cada

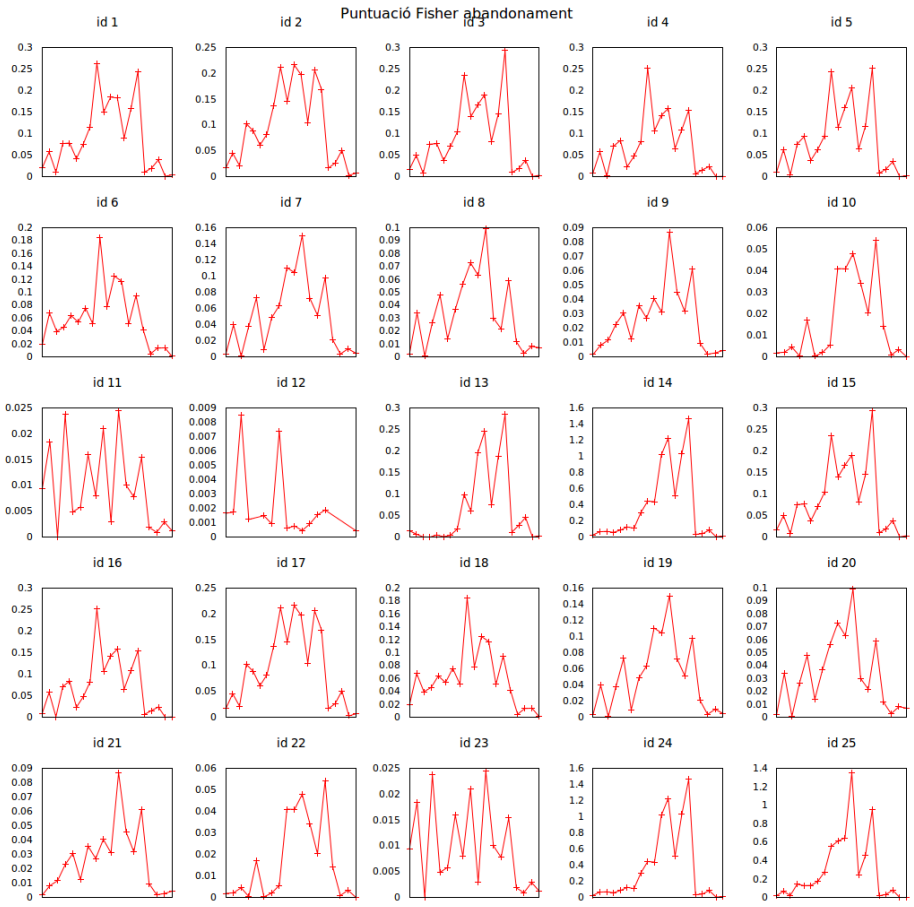


Figura 12: Puntuació Fisher (eix y) per cadascuna de les 25 característiques descrites a la taula 18 i cada setmana (eix x) respecte al abandó del curs.

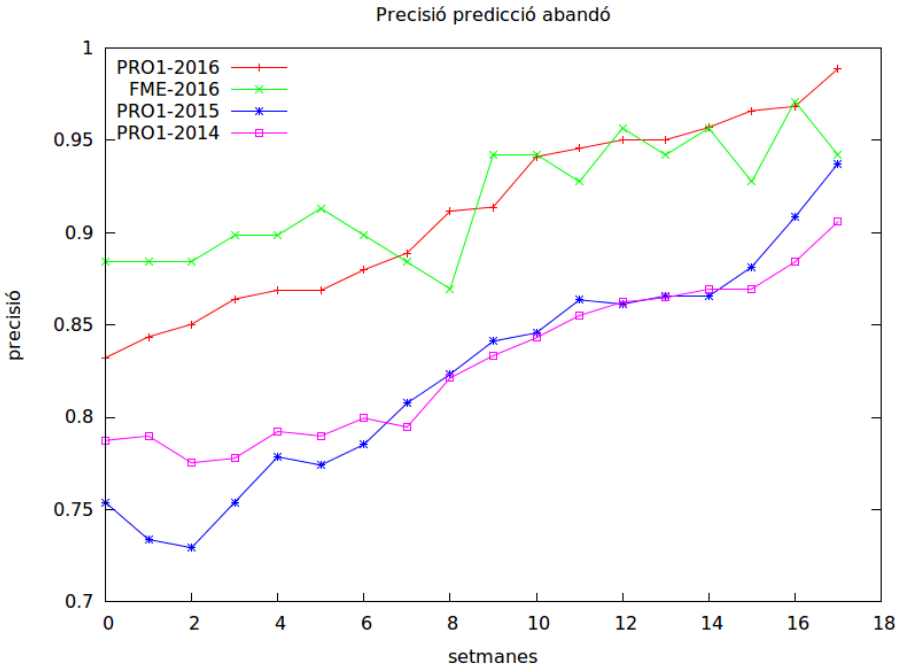


Figura 13: Precisió de la predicció d'abandó per a cada curs. S'ha de tenir en compte que el SVM ha estat entrenat amb el curs PRO1-2016 i per tant representa la precisió de training.

setmana.

La gràfica 13 mostra els resultats obtinguts. Per tenir un punt de referència a l'hora de saber com de bons són els resultats, la gràfica 14 compara el predictor amb un de constant que prediu el mateix per a tots els estudiants.

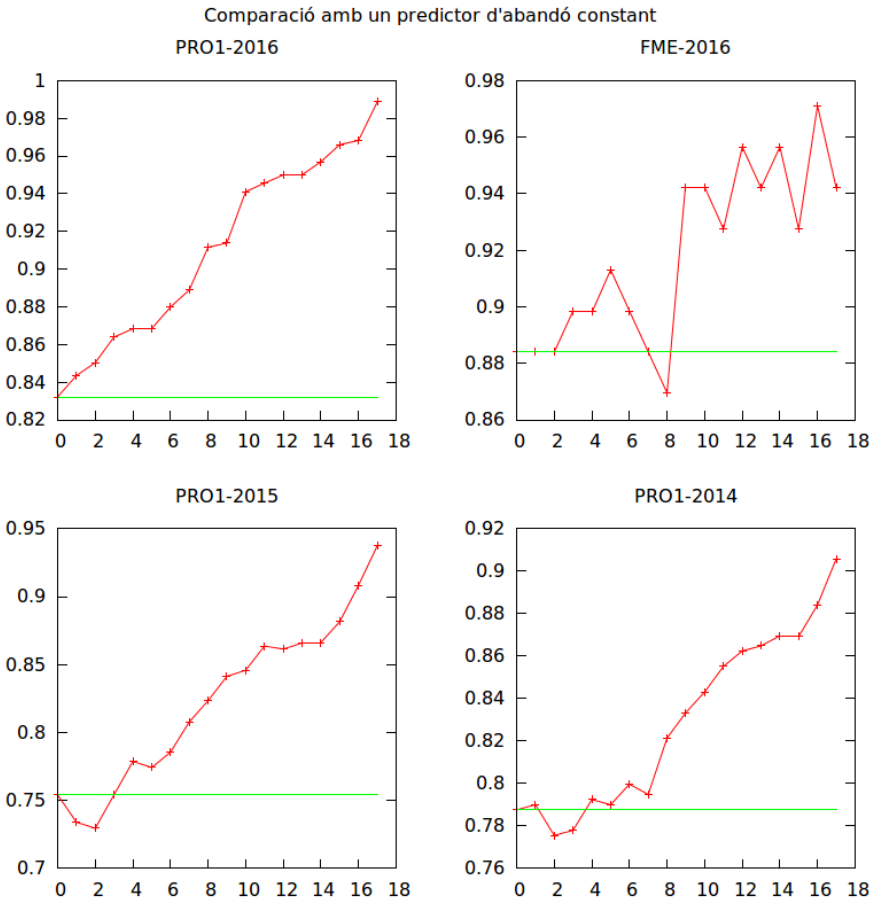


Figura 14: Comparació entre la precisió (eix y) del predictor d'abandó (vermell) i un de constant (verd) cada setmana (eix x).

## 8.4 Predicció de la nota final

Considerem que les notes dels estudiants són influenciades en molts casos per factors que no podem controlar com: subjectivitat dels correctors, sort i còpies, és a dir, que la diferència entre un 1 i un 3 no està clara. Això dificulta la tasca de predir la nota exacta dels estudiants.

Per solucionar aquesta consideració hem optat per no predir la nota exacta dels estudiants i convertir el problema en un de classificació, agrupant els estudiants en funció de la seva nota d'acord a la funció:

$$c = \begin{cases} 1 & \text{si nota} = 0 \\ \left\lceil \frac{3\text{nota}}{10} \right\rceil & \text{altrament} \end{cases}$$

D'aquesta manera aconseguim agrupar els estudiants en 3 classes en funció de la seva nota:

| Classe | Nota | | |
|---|---|---|---|
| 1 | 0 | – | 3.33 |
| 2 | 3.33 | – | 6.66 |
| 3 | 6.66 | – | 10 |

Com que ara tenim més de dos classes, apliquem la generalització de la puntuació de Fisher (9):

$$F(\boldsymbol{x}^j) = \frac{\sum_{c=1}^{k} (\mu_c^j - \mu^j)^2}{\sum_{c=1}^{k} n_c (\sigma_c^j)^2} \tag{10}$$

La figura 15 mostra els resultats obtinguts.

Esperàvem que els veredictes (id 6–11), especialment el nombre d'AC (id 6), tinguessin una gran rellevància; però ha resultat que el nombre d'enviaments (id 4) puntua molt més alt. És possible que els veredictes puntuïn en conjunt més alt, però la nostra versió de Fisher només té en compte característiques de forma individual [8].

Podem veure a la figura 15 que el nombre d'enviaments dolents (id 19) puntua més alt que el nombre d'AC per dia (id 18) a partir de mig curs. Tenim la hipòtesis de que això és culpa de que les solucions estan penjades a pagines web externes com pastebin. Els estudiants es copien d'allà i per això tenir un AC no és tant significatiu com haver-ho intentat i fallat.

També es pot observar que el nombre de d'enviament (id 4 i 5) tendeixen a obtenir valors alts a totes les setmanes. Això indica que la constància amb Jutge.org influeix significativament en les notes.

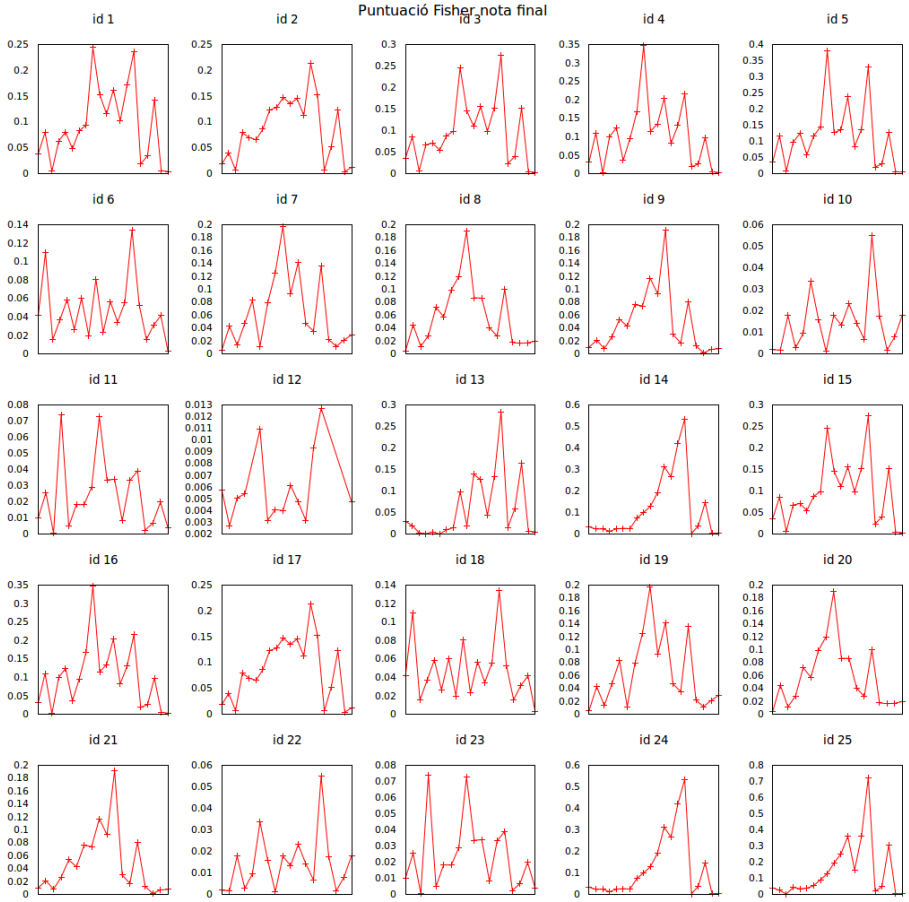


Figura 15: Puntuació Fisher (eix y) per cadascuna de les 25 característiques descrites a la taula 18 i cada setmana (eix x) respecte a la nota final.

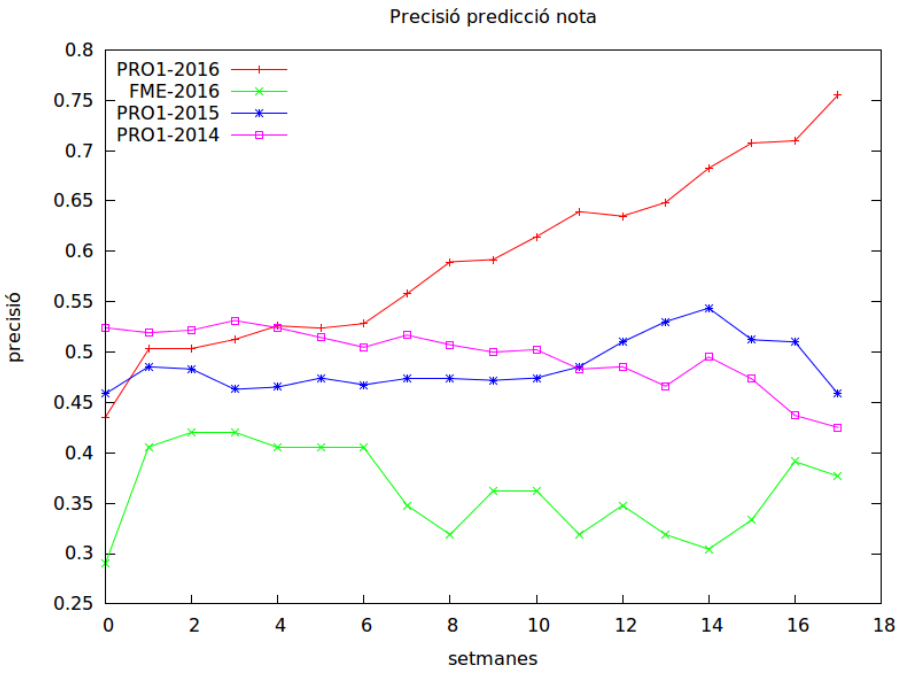


Figura 16: Precisió de la predicció de la nota final per a cada curs. S'ha de tenir en compte que el SVM ha estat entrenat amb el curs PRO1-2016 i per tant representa la precisió de training.

Pels motius esmentats al apartat anterior hem escollit les hem escollit les 5 característiques amb més puntuació de Fisher: Actiu2, Nombre de sessions per dia, Actiu1, Nombre d'enviaments consultats i Nombre d'enviaments.

Hem utilitzat el mateix SVM calibrant-lo de la manera descrita anteriorment: "grid-tunning" 5-fold cross-validation amb el curs PRO1-2016. Finalment, hem obtingut $C = 32$ i $\gamma = 2^{-13}$.

De la mateixa manera hem utilitzat el curs PRO1-2016 per entrenar l'algoritme i l'hem avaluat contra els altres cursos.

La gràfica 16 mostra els resultats obtinguts. Hem comparat el predictor amb un de constant que prediu la millor classe a tots els estudiants, és a dir, si $p(c)$ és la precisió d'un predictor que sempre prediu la classe $c$, aquest predictor constant escull $c$ de manera que $p(c)$ sigui màxim.

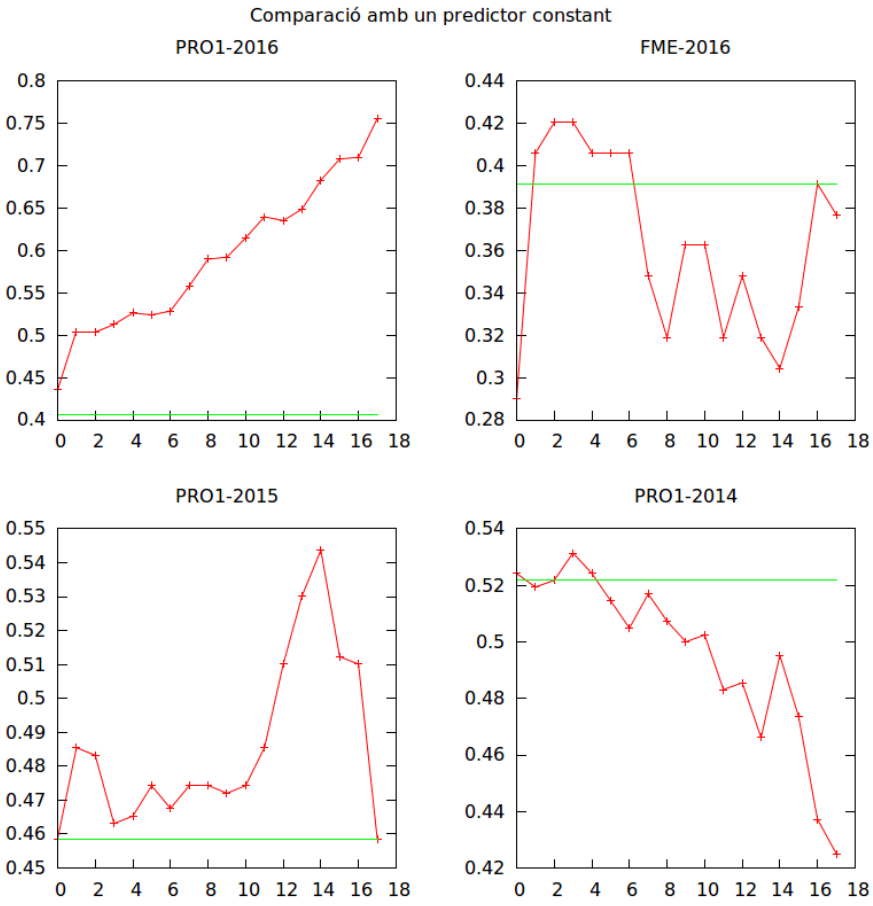


Figura 17: Comparació de la precisió (eix y) del predictor de nota final (vermell) i un predictor constant (verd) cada setmana (eix x).

## 8.5 Valoració dels resultats obtinguts

A la figura 14 podem veure que en la predicció d'abandó s'han obtingut bons resultats.

Els predictors comencen amb la mateixa precisió que predir que tothom aprova, però cap a la setmana 5 ja donen millors resultats. En general aconseguim una millora de fins al 15 % comparat amb el predictor constant.

El curs PRO1-2014 no té el camp sessió definit i per tant totes les seves sessions són fantasma. Creiem que per això aquest curs mostra una millora menys pronunciada.

El curs FME-2016 mostra un comportament diferent als altres cursos. Tenint en compte que el SVM s'ha entrenat, i calibrat, amb un curs de programació de la FIB, pensem que es deu a diferencies entre les facultats o el comportament dels estudiants d'aquestes.

En la predicció de la nota els resultats obtinguts han estat inferiors als esperats. Les gràfiques 16 i 17 mostren que el predictor ens ha quedat sobre-ajustat en favor del curs PRO1-2016. Segurament ha estat per haver calibrat i entrenat només amb el curs PRO1-2016. Queda pendent per al futur re-calibrar i entrenar el SVM amb altres dades.

A la gràfica 17 veiem que el curs PRO1-2015 mostra un comportament similar, però molt més moderat, a PRO1-2016 des de la setmana 4 fins la 14. Aquesta caiguda que mostra a partir de la setmana 15, ens fa pensar especialment en el sobre-ajustament, i en diferències entre els cursos.

Altre cop el curs de matemàtiques, FME-2016, mostra un comportament irregular.

# 9 Creació d'un recomanador

## 9.1 Descripció del problema

Jutge.org té molts problemes. Cada problema té una dificultat concreta i tracta una tema concret (recursivitat, sub-funcions, matrius ...).

Principalment, els usuaris estan interessats en dos aspectes: aprofundir en el tema que estan tractant i saltar a un nou tema. Però Jutge.org no ofereix al seus usuaris cap manera per poder fer cap de les dues coses. Els usuaris es veuen obligats a buscar problema per problema fins que troben un en el que estan interessats.

En aquest treball volem crear un recomanador de problemes que sigui capaç de guiar als estudiants: per una banda aprofundint en els temes d'estudi de l'estudiant i per altra banda proposant temes nous.

Aquesta secció detalla la construcció del recomanador de problemes per a Jutge.org.

## 9.2 Descripció del recomanador

Per restriccions temporals s'ha dissenyat un recomanador basat en contingut que utilitza la dificultat dels problemes, estimada a la secció 7, com a base per fer les recomanacions.

Hem decidit construir el recomanador utilitzant tres dels algoritmes presentats a l'apartat 7.2: Naïf, JutgeRank i SVD2. Hem descartat SVD1 per ser igual a SVD2. Suposem que JutgeRank obtindrà millors resultats perquè assigna les dificultats de forma més dispersa.

A partir de les mesures de dificultat normalitzades dels problemes, el recomanador construeix una matriu de diferències entre problemes, $\boldsymbol{R}^{P\times P}$, on cada posició de $\boldsymbol{R}$ ve donada per:

$$\boldsymbol{R}_{ij} = dif(i,j) \tag{11}$$

Hem decidit definir la diferència entre problemes de manera simètrica tal que:

$$dif(i,j) = |\boldsymbol{d}_i - \boldsymbol{d}_j|$$

on $\boldsymbol{d}$ és la dificultat normalitzada dels problemes, calculada per qualsevol dels algoritmes descrits al apartat 7.2.

Donada la matriu $\boldsymbol{M}$, la matriu de diferències $\boldsymbol{R}$ i un usuari $j$, el recomanador retorna el problema amb menor diferència als problemes fets per l'usuari $j$.

Utilitzant la matriu $\boldsymbol{R}$ i el vector de problemes $\boldsymbol{u}_j$ es calcula el vector de diferencies als problemes fets: (Recordem que $\boldsymbol{u}_j$ representa la columna $j$ de $\boldsymbol{M}$, és a dir, un vector dels problemes fets per l'usuari $j$)

$$\boldsymbol{r} = \boldsymbol{R} \cdot \boldsymbol{u}_j \tag{12}$$

on $\boldsymbol{r}$ representa la diferència d'un problema qualsevol respecte a tots els problemes que ha resolt un usuari qualsevol.

Per evitar recomanar problemes ja resolts cada element $\boldsymbol{r}_i \in \boldsymbol{r}$ es transforma d'acord a:

$$\boldsymbol{r}_i = \begin{cases} \infty & \text{Si el problema } i \text{ ha estat resolt per l'usuari} \\ \boldsymbol{r}_i & \text{en cas contrari} \end{cases} \tag{13}$$

D'aquesta manera els problemes ja resolts tenen diferència màxima.
Finalment, es recomana el primer problema amb diferència mínima.

### Exemple

En aquest apartat es detalla l'aplicació del recomanador utilitzant l'algoritme JutgeRank i la matriu d'exemple descrita a l'apartat 7.2.

Calculada per JutgeRank la dificultat dels problemes d'exemple venia donada per:

$$\boldsymbol{d} = \begin{bmatrix} 0.37 \\ 0.48 \\ 0.32 \\ 0.38 \\ 0.44 \\ 0.45 \end{bmatrix}$$

Després de normalitzar la dificultat apliquem l'equació (11) i trobem la matriu de diferències:

$$\boldsymbol{R} = \begin{pmatrix} 0.00 & 0.11 & 0.04 & 0.01 & 0.07 & 0.08 \\ 0.11 & 0.00 & 0.15 & 0.10 & 0.04 & 0.03 \\ 0.04 & 0.15 & 0.00 & 0.05 & 0.11 & 0.13 \\ 0.01 & 0.10 & 0.05 & 0.00 & 0.06 & 0.08 \\ 0.07 & 0.04 & 0.11 & 0.06 & 0.00 & 0.01 \\ 0.08 & 0.03 & 0.13 & 0.08 & 0.01 & 0.00 \end{pmatrix}$$

Supose'm que volem recomanar un problema al usuari 2, que ha fet els problemes 2, 3 i 5.

Primer calculem el seu vector de diferències $\boldsymbol{r}$, aplicant l'equació (12) trobem:

$$\boldsymbol{r} = \begin{bmatrix} 0.22 \\ 0.19 \\ 0.26 \\ 0.21 \\ 0.15 \\ 0.17 \end{bmatrix}$$

Notem que:

$$\begin{bmatrix} 0.22 \\ 0.19 \\ 0.26 \\ 0.21 \\ 0.15 \\ 0.17 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{1,2} + \boldsymbol{R}_{1,3} + \boldsymbol{R}_{1,5} \\ \boldsymbol{R}_{2,2} + \boldsymbol{R}_{2,3} + \boldsymbol{R}_{2,5} \\ \boldsymbol{R}_{3,2} + \boldsymbol{R}_{3,3} + \boldsymbol{R}_{3,5} \\ \boldsymbol{R}_{4,2} + \boldsymbol{R}_{4,3} + \boldsymbol{R}_{4,5} \\ \boldsymbol{R}_{5,2} + \boldsymbol{R}_{5,3} + \boldsymbol{R}_{5,5} \\ \boldsymbol{R}_{6,2} + \boldsymbol{R}_{6,3} + \boldsymbol{R}_{6,5} \end{bmatrix}$$

Si recomanéssim ara, el problema amb diferència mínima, hauríem de recomanar el problema 5. Però estaríem recomanant un problema que l'usuari ja ha completat. Per això apliquem la transformació (13) per obtenir el vector $\boldsymbol{r}$ que ens interessa:

$$\boldsymbol{r} = \begin{bmatrix} 0.22 \\ \infty \\ \infty \\ 0.21 \\ \infty \\ 0.17 \end{bmatrix}$$

Ara el problema amb la diferència mínima és el problema 6, i aquest és el que el recomanador retorna.

Podem comprovar amb les dificultats dels problemes, que realment se està recomanant el problema més similar.

## 9.3 Validació

### 9.3.1 Descripció del mètode de validació

Assumim que un usuari només completa els problemes en els que està interessat. Segons això, si marquem un problema realitzat com a no realitat (extraiem un problema) el recomanador hauria de recomanar aquest problema.

S'ha implementat un sistema que avalua si el recomanador recomana els problemes extrets, i calcula l'error com la diferència al quadrat entre la dificultat del problema extret i la dificultat del problema recomanat.

Es fa una recomanació a cada usuari i es fa el promig dels errors. Per poder tenir un error en unitats de dificultat es calcula l'arrel de l'error quadràtic mig. Sigui $\boldsymbol{a}_u$ el problema extret al usuari $u$ i $\boldsymbol{b}_u$ el problema extret a aquest mateix usuari:

$$\text{Error} = \sqrt{\frac{1}{U}\sum_{u=1}^{U}(\boldsymbol{d}_{\boldsymbol{a}_u} - \boldsymbol{d}_{\boldsymbol{b}_u})^2} \tag{14}$$

Només hem extret un problema per usuari però ho hem fet de forma aleatòria sobre els problemes completats. Finalment el procés de calcul de l'error s'ha executat 10 cops i hem calculat la mitjana dels errors donats.

El procés de validació queda resumit en:

1. Per a cada usuari: extreure un problema al atzar dels problemes realitzats.

2. Recomanar un problema a cada usuari.

3. Aplicar (14) per trobar l'error.

4. Repetir 10 cops i calcular al mitjana dels errors.

### 9.3.2 Validació

Per simplificar la interpretació dels errors hem calculat quants problemes entren dintre del marge d'error del recomanador, és a dir, calculem el nombre de problemes amb diferència menor a l'error.

La taula següent mostra els resultats obtinguts:

| Algoritme | Curs | Error | Problemes | |
|---|---|---|---|---|
| JutgeRank | PRO1-2016 | 0.51 | 90 | (29.56 %) |
| | FME-2016 | 0.66 | 99 | (49.23 %) |
| | PRO1-2015 | 0.44 | 82 | (27.68 %) |
| | PRO1-2014 | 0.57 | 90 | (30.58 %) |
| Naïf | PRO1-2016 | 0.98 | 174 | (57.24 %) |
| | FME-2016 | 0.64 | 83 | (41.25 %) |
| | PRO1-2015 | 0.84 | 163 | (54.90 %) |
| | PRO1-2014 | 0.86 | 162 | (54.84 %) |
| SVD2 | PRO1-2016 | 0.93 | 162 | (53.18 %) |
| | FME-2016 | 0.60 | 83 | (41.06 %) |
| | PRO1-2015 | 0.79 | 156 | (52.56 %) |
| | PRO1-2014 | 0.79 | 139 | (47.06 %) |

Taula 19: Error del recomanador (en unitats de dificultat normalitzades) amb la dificultat mesurada per diferents algoritmes. La columna problemes mostra el nombre de problemes amb diferència més petita que l'error i la proporció respecte al nombre de problemes del curs, és a dir, el nombre de problemes que representa l'error.

El rendiment del recomanador no és tan positiu com s'esperava. En el millor dels escenaris, hi ha 82 problemes entre el problema extret i el recomanat.

Es possible que part del problema estigui en el mètode de validació escollit. Aquest mètode no té en compte que els usuaris volen sortir del tema d'estudi i això els porta fer problemes més fàcils.

També hem pogut comprovar que JutgeRank obté millors resultats. Els altres algoritmes assignen dificultats massa similars.

Altre cop destaca el curs de matemàtiques, FME-2016. Ens els altres cursos, JutgeRank ha aconseguit els millors resultats, però els tres algoritmes han obtingut resultats similars per al curs FME-2016.

# 10 Conclusions

Els jutges de programació contenen informació útil sobre els seus usuaris amagada entre les seves dades. En aquest projecte hem analitzat la base de dades i els logs d'accés d'un jutge de programació, Jutge.org, per obtenir-les.

Per poder validar els resultats, el projecte s'ha centrat en quatre cursos per als que es disposaven dades. Sobre aquests cursos s'ha calculat la dificultat dels problemes i l'experiència dels estudiants; també s'ha entrenat un algoritme d'aprenentatge automàtic per predir els estudiants que abandonen un curs i la seva nota. Finalment, també hem presentat un recomanador de problemes.

Hem observat que els alumnes del curs de programació de la facultat de matemàtiques i estadística (FME) obtenen millors notes que els alumnes de la facultat d'informàtica (FIB) tot i que la diferència s'està reduint. Aquesta diferència entre el curs de la FME i els de la FIB ha provocat que els algoritmes es comportessin de forma diferent entre les facultats. Això ens suggereix que *en un futur s'hauria de separar l'anàlisi dels cursos de diferents facultats.*

En aquest treball proposem quatre algoritmes per calcular la dificultat dels problemes i l'experiència dels estudiants, però observem que no hi ha diferències significatives entre ells a l'hora de calcular l'experiència dels usuaris. En canvi, en el càlcul de la dificultat del problemes, JutgeRank obté resultats diferents. Això ens suggereix que es podria millorar el càlcul de l'experiència en JutgeRank.

Hem analitzat els logs d'accés de Jutge.org i hem extret informació del comportament del seus usuaris. Amb aquestes dades hem entrenat un Support Vector Machine (SVM) per predir l'abandó i hem obtingut bons resultats.

Després hem intentat predir la nota final, però ens ha quedat sobre-ajustat i hem obtingut uns resultats dolents. Segurament, el motiu ha estat haver calibrat i entrenat el SVM amb un curs concret. Comprovar si realment és aquest el motiu queda pendent per al futur.

Hem observat que *el nombre de solucions correctes influeix menys en la nota final que el nombre de solucions incorrectes.* Pensem que això estaria relacionat amb els alumnes copiant les solucions sense intentar resoldre els problemes.

Finalment, hem construït un recomanador de problemes utilitzant les mesures de dificultat calculades anteriorment, però els resultats no han estat prou satisfactoris i considerem que s'hauria de millorar en un futur.

Tota la part tècnica del treball ha estat realitzada en Julia. Hem pogut comprovar que Julia té una sintaxis senzilla que ha simplificat en molts casos el desenvolupament del projecte. Ens ha agradat treballar en Julia i estem desitjant tornar-ho a fer.

Aquest treball ha servit per posar de manifest el potencial dels algoritmes d'aprenentatge automàtic per millorar la qualitat i la eficàcia de les docències.

# Referències


[1] Andres Alvarez i Terry A. Scott. "Using Student Surveys in Determining the Difficulty of Programming Assignments". A: *J. Comput. Sci. Coll.* 26.2 (des. de 2010), pàg. 157 - 163. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=1858583.1858605.

[2] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006. 738 pàg. ISBN: 978-0-387-31073-2.

[3] Carla D. Martin i Mason A. Porter. "The Extraordinary SVD". A: *The American Mathematical Monthly* 119.10 (des. de 2012), pàg. 838 - 851. ISSN: 00029890, 19300972. URL: http://www.jstor.org/stable/info/10.4169/amer.math.monthly.119.10.838 (cons. 4-5-2016).

[4] *Codde Abbey*. URL: http://www.codeabbey.com.

[5] Chih-Wei Hsu, Chih-Chung Chang i Chih-Jen Lin. *A Practical Guide to Support Vector Classification*. Mai. de 2016. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[6] Gerben Dekker, Mykola Pechenizkiy i Jan Vleeshouwers. "Predicting students drop out: A case study". A: *Educational Data Mining 2009*. 2009. URL: http://www.educationaldatamining.org/conferences/index.php/EDM/2009/paper/download/1467/1433 (cons. 8-10-2016).

[7] Neha Goel i C. K. Jha. "Analyzing users behavior from web access logs using automated log analyzer tool". A: *International Journal of Computer Applications* 62.2 (2013). URL: http://search.proquest.com/openview/e11992a75cfb298b506608307cdd49a5/1?pq-origsite=gscholar (cons. 28-9-2016).

[8] Quanquan Gu, Zhenhui Li i Jiawei Han. "Generalized fisher score for feature selection". A: *arXiv preprint arXiv:1202.3725* (2012). URL: http://arxiv.org/abs/1202.3725 (cons. 24-5-2016).

[9] Hong Wei Ma, Guang Wei Zhang i Peng Li. "Survey of Collaborative Filtering Algorithms". A: *Journal of Chinese Computer Systems*. 2009, pàg. 1282 - 1288.

[10] *How are the problem difficulty and the author rating calculated?* Frequently Asked Questions @ Timus Online Judge. URL: http://acm.timus.ru/help.aspx?topic=faq (cons. 1-4-2016).

[11] *How much carbon dioxide is produced per kilowatthour when generating electricity with fossil fuels? - FAQ - U.S. Energy Information Administration (EIA)*. URL: https://www.eia.gov/tools/faqs/faq.cfm?id=74&t=11 (cons. 7-10-2016).

[12] *Informe de sostenibilitat del TFG*. URL: http://www.fib.upc.edu/fib/estudiar-enginyeria-informatica/treball-final-grau/mainColumnParagraphs/013/document/2016_SyCS_GEP_v3_catala.pdf (cons. 5-10-2016).

[13] *Jutge.org - Documentation*. URL: https://jutge.org/documentation/verdicts (cons. 25-9-2016).

[14] *Kendall tau distance*. A: *Wikipedia, the free encyclopedia*. Page Version ID: 723196467. 1 de jun. de 2016. URL: https://en.wikipedia.org/w/index.php?title=Kendall_tau_distance&oldid=723196467 (cons. 9-10-2016).

[15] Khairil Imran Ghauth i Nor Aniza Abdullah. *Measuring learner's performance in e-learning recommender systems*. 2010.

[16] Marius Kloft et al. "Predicting MOOC dropout over weeks using machine learning methods". A: *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*. 2014, pàg. 60 - 65. URL: http://www.aclweb.org/anthology/W/W14/W14-41.pdf#page=67 (cons. 16-5-2016).

[17] *k-means clustering*. A: *Wikipedia, the free encyclopedia*. Page Version ID: 735453167. 20 d'ago. de 2016. URL: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=735453167 (cons. 1-4-2016).

[18] Greg Linden, Brent Smith i Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering". A: *Internet Computing, IEEE* 7.1 (2003), pàg. 76 - 80. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1167344 (cons. 1-4-2016).

[19] *Machine learning*. A: *Wikipedia, the free encyclopedia*. Page Version ID: 742249378. 2 d'oct. de 2016. URL: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=742249378 (cons. 5-10-2016).

[20] "Method for node ranking in a linked database". US6285999 B1. Lawrence Page. U.S. Classification 1/1, 715/207, 715/206, 707/E17.108, 715/230, 707/E17.097, 715/256, 707/999.005, 707/999.007; International Classification G06F17/30; Cooperative Classification Y10S707/99935, Y10S707/99937, G06F17/30864, G06F17/30728; European Classification G06F17/30W1, G06F17/30T6C. 4 de set. de 2001. URL: http://www.google.com/patents/US6285999 (cons. 9-10-2016).

[21] Roger Meyer i Carlos Cid. "Detecting attacks on web applications from log files". A: *Sans Institute, InfoSec reading room* (2008).

[22] Michael D. Ekstrand, John T. Riedl i Joseph A. Konstan. "Collaborative Filtering Recommender Systems". A: *Foundations and Trends(R) in Human–Computer Interaction*. Vol. 4. 2. 2010, pàg. 81 - 173.

[23] Adam Oliner i Jon Stearley. "What Supercomputers Say: A Study of Five System Logs". A: IEEE, jun. de 2007, pàg. 575 - 584. ISBN: 978-0-7695-2855-7. URL: http://ieeexplore.ieee.org/document/4273008/.

[24] Ruoming Pang et al. "Characteristics of Internet Background Radiation". A: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. IMC '04. Taormina, Sicily, Italy: ACM, 2004, pàg. 27 - 40. ISBN: 1-58113-821-0. URL: http://doi.acm.org/10.1145/1028788.1028794.

[25] Jordi Petit, Omer Giménez i Salvador Roura. "Jutge.Org: An Educational Programming Judge". A: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. SIGCSE '12. New York, NY, USA: ACM, 2012, pàg. 445 - 450. ISBN: 978-1-4503-1098-7. URL: http://doi.acm.org/10.1145/2157136.2157267 (cons. 1-4-2016).

[26] *Precio neto de la electricidad para uso doméstico y uso industrial - IV_ 12.pdf*. URL: http://www.minetur.gob.es/es-ES/IndicadoresyEstadisticas/DatosEstadisticos/IV.%20Energ%C3%ADa%20y%20emisiones/IV_12.pdf (cons. 31-3-2016).

[27] *Project Euler*. URL: https://projecteuler.net (cons. 1-4-2016).

[28] Rodrigo Elias Francisco i Ana Paula Ambrosio. *Mining an Online Judge System to Support Introductory Computer Programming Teaching*. 2015.

[29] Ruiguo Yu et al. *The Research of the Recommendation Algorithm in Online Learning*. 2015. URL: http://dx.doi.org/10.14257/ijmue.2015.10.4.08.

[30] *Scrum Guide*. URL: http://www.scrumguides.org/ (cons. 11-10-2016).

[31] *The Julia Language*. URL: http://julialang.org/ (cons. 8-10-2016).

[32] *Timus Online Judge*. URL: http://acm.timus.ru.

[33] *Wiki: Ranks - CodeAbbey*. URL: http://www.codeabbey.com/index/wiki/ranks (cons. 1-4-2016).

[34] Xindong Wu et al. "Top 10 algorithms in data mining". A: *Knowledge and Information Systems* 14.1 (gen. de 2008), pàg. 1 - 37. ISSN: 0219-1377, 0219-3116. URL: http://link.springer.com/10.1007/s10115-007-0114-2 (cons. 1-4-2016).